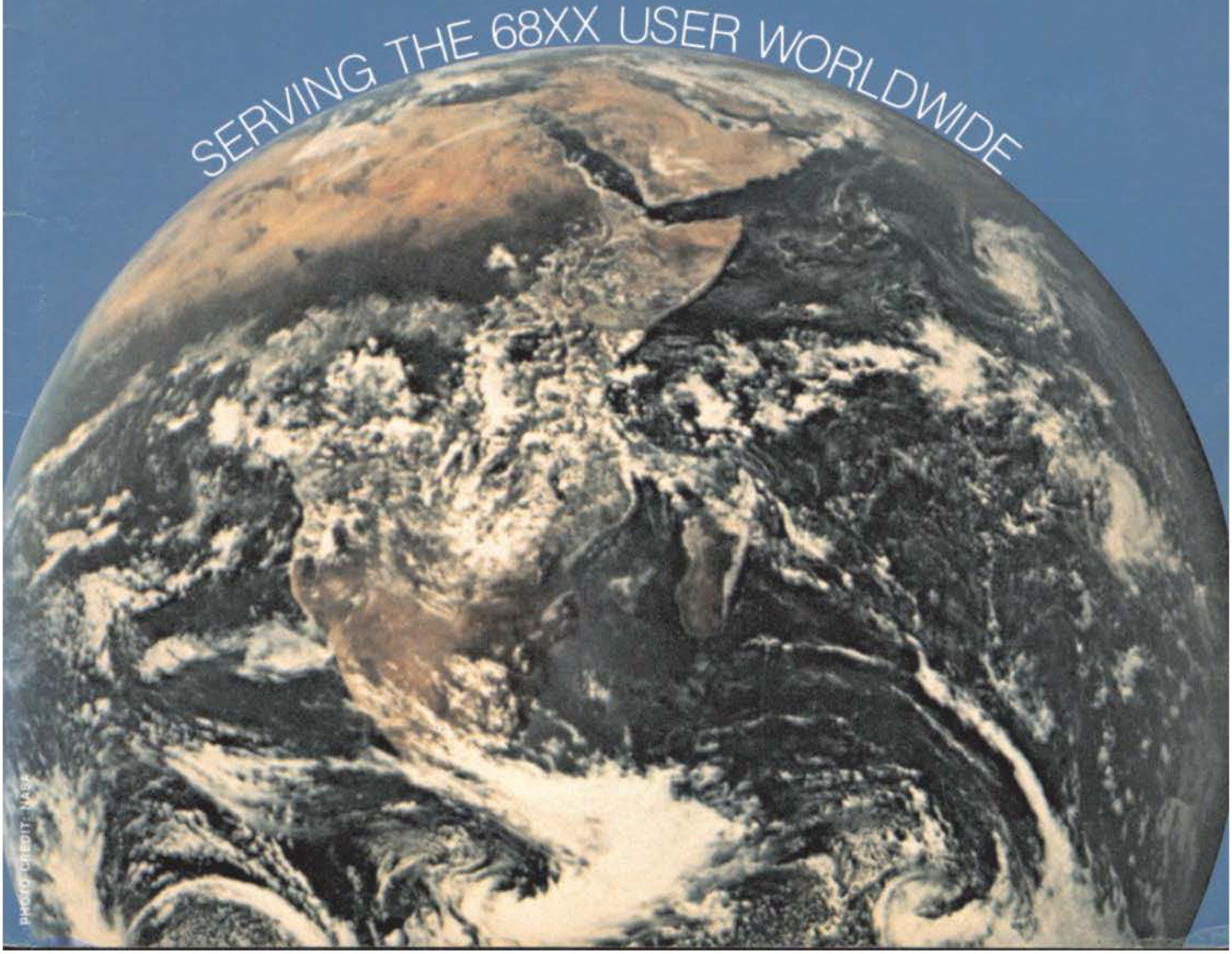
'68' MICRO JOURNAL

$2.95 USA

Australia	A $4.75	New Zealand	NZ $ 6.50
Singapore	S $9.45	Hong Kong	H $23.50
Malaysia	M $9.45	Sweden	30:-SEK

VOLUME V ISSUE VII • Devoted to the 68XX User • July 1983

"Small Computers Doing Big Things"



YOUR CHOICE-smart either way

- Over 140 software driven functions
- 82 x 24 or 82 x 20 screen format — software selectable
- High resolution 7 x 12 matrix characters — P-31 green phosphor
- Upper/lower case character set — plus graphics character set
- 56-key alphanumeric keyboard — plus 12-key cursor, numeric pad
- Internal editing functions — insert, delete, scroll, roll, slide, etc.
- Parallel printer I/O port
- 50 to 38,400 baud operation — programmable
- Cursor type, cursor position, print control characters, protected fields, shift inversion, dual intensity and many other features

8212 — twelve-inch diagnonal screen or 8209 — nine-inch diagnonal screen

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

The UniFLEX™ Operating System extracts every last drop

from the 8 bit 6809 microprocessor allowing it to outperform many 16 bit systems

With the UniFLEX™ Operating System, the 8 bit 6809 microprocessor can perform as well as larger CPUs in a multi-user, multi-tasking environment.

Independently developed from the ground up, UniFLEX™ closely models the features found in the UNIX™ Operating System. And in two years of use, UniFLEX™ has proven the abilities of the 6809 to perform large system functions when incorporated into a properly designed mainframe.

Some of the features supported include:

- full multi-user, multi-tasking capabilities
- hierarchical file systems
- device independent I/O
- four Gigabyte disk capacities
- full file protection
- inter-task communication via pipes
- I/O redirection
- taskswapping for efficient memory usage
- full random-access files
- comprehensive shell command language
- foreground-background jobs
- electronic mail and printer spooling
- system accounting facilities

The support software currently available for use under UniFLEX™ is extensive. A sampling of the programs available includes:

- native C compiler (full implementation)
- native Pascal compiler
- FORTRAN 77 ANSI Subset compiler
- COBOL compiler with ISAM files, Report Writer & Sort/Merge
- Extended BASIC interpreter
- Extended BASIC precompiler
- text editing and processing software
- enhanced printer spooler
- variety of absolute and relocatable assemblers
- debug and diagnostic packages

Technical Systems Consultants, Inc. also offers a line of single user FLEX™ software products for 6800 and 6809 processors. For those having an absolute need for a 16 bit processor, UniFLEX™ will be available through OEM licensing arrangements for the 68000 microprocessor. Please call or write for additional information on individual products or OEM licensing arrangements.

UNIX™ is a trademark of Bell Laboratories.
FLEX™ and UniFLEX™ are trademarks of Technical Systems Consultants, Inc.

technical systems consultants, inc.

111 Providence Road
Chapel Hill, North Carolina 27514
(919) 493-1451

'68' MICRO JOURNAL

Portions of the text for 68 MICRO JOURNAL was prepared using the folowing furnished hard/software.

COMPUTERS-HARDWARE
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216
S09-5/8 DMF disk-CDS1-8212W-Sprint 3 Printer

GIMIX Inc.
1337 West 37th Place
Chicago, IL 60609
Super Mainframe-OS9-FLEX-Assorted Hardware

EDITORS-WORD PROCESSORS
Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514
FLEX-Editor-Processor

Great Plains Computer Company, Inc.
PO Box 916
Idaho Falls, ID 83401
STYLO-Mail Merge

Editorial Staff

Don Williams Sr.	Publisher
Larry E. Williams	Executive Editor
Tom E. Williams	Production Editor
Robert (Bob) Nay	Color Editor

Administrative Staff

Mary Robertson	Office Manager
Joyce Williams	Accounting
Carolyn Williams	Subscriptions
Penny Williams	File Management

Contributing Editors

Ron Anderson
Norm Commo
Peter Dibbie
Dr. Theo Elbert
William E. Fisher
Dr. E.M. Pass

Special Technical Projects

Cley Abrams K6AEP
Tom Hunt

CONTENTS

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
615 842-4600

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tenn. and additional entries. Postmaster: send Form 3579 to 68' Micro Journal, PO Box 849, Hixson, Tennessee.

SUBSCRIPTION RATES
USA
1-Year $24.50 2-Years $42.50 3-Years $64.50
FOREIGN
See Page 52

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors full **name, address, date and telephone number.** It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

The following TSC Text Processor commands **ONLY** should be used (due to our proportional processor): **.sp** space, **.pp** paragraph, **.fi** fill and **.nf** no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the **.pg** page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX™ format, either FLEX2 6800, or FLEX9 6809 any version.

If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please use a dark ribbon!**

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste'.

Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

The GMX 6809 CPU III

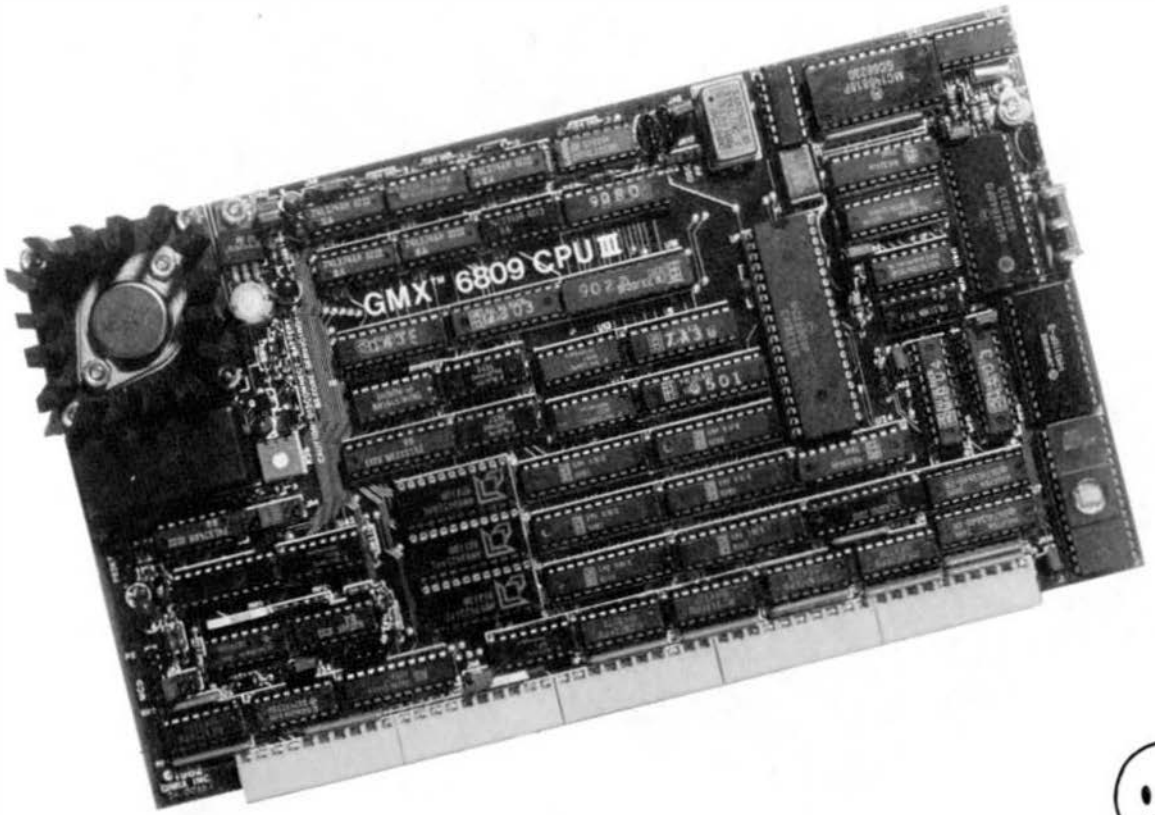
The GIMIX 6809 CPU III board is an advanced design specifically intended for use with multi-user, multi-tasking operating systems.

Built on a multi-layer circuit board and utilizing high-speed, high-density logic, the GMX CPU III enhances the performance of the 2MHz 68B09 by providing such features as 1 byte/microsecond DMA block transfers from memory to memory or between memory and I/O devices, and advanced memory management with 2K segments and segment attributes. The board automatically arbitrates DMA contention between the on board DMA and external DMA devices such as disk controllers. The 2K memory segments allow more efficient memory usage. The segment attributes allow the trapping of out-of-range memory references, write protection, and a hardware single step function for software debugging.

The board prevents the execution of certain illegal instructions from crashing the system by monitoring interrupts to the 6809 and its response to them. If the processor does not respond to an interrupt within 128 clock cycles the board resets the 6809 and asserts a special reset vector. The system can then close down the offending task and resume normal operation. This also limits the length of time that interrupts can remain masked by a user, preventing users from keeping the system from task switching and servicing other users.

To further protect the system, the CPU board supports separate user and system "states" with automatic switching to the system state in response to interrupts and system (SWI) calls. Certain functions and memory areas can only be accessed in the system state, preventing unauthorized accesses.

The GMX CPU III also includes a full function time-of-day clock with year and automatic leap year/daylight savings time correction, and a 2K scratchpad RAM; both with battery back-up. To provide precision timing functions, a 6840 PTM with a separate 500 KHz. precision (.0025%) time base oscillator is included. The oscillator is easily user replaceable to provide other time base frequencies (750 KHz. maximum). The single EPROM socket will accept 2K, 4K or 8K EPROMS, with a maximum of 4K mapped into the system address space at any one time. Software switching is implemented by selecting the upper or lower half of an 8K EPROM under hardware or software control.



OS-9 GMX III Operating System

OS-9 GMX III is an enhanced OS-9 Level II that takes full advantage of the features of the GMX CPU III. As a result, the system is faster, more memory efficient, and a more secure multi-user, multi-tasking operating system than the original OS-9 GIMIX II, while retaining complete software compatibility. Throughput is enhanced by the memory to memory DMA and the automatic task switching, while the memory attributes and illegal instruction trapping protect the system and individual users from each other. Sharable system modules in RAM are write protected to prevent tampering. Memory mapping in 2K segments and the ability to load modules in non-contiguous RAM provide more efficient memory utilization. Each task can be allocated a full 64K of RAM, with no operating system overhead in the tasks address space.

UniFLEX for the GMX 6809 CPU III and Intelligent I/O boards is in development.

GMX INC. 1337 West 37th Place • Chicago, Illinois 60609 • (312) 927-5510 • TWX 910-221-4055

FLEX™ USER NOTES
THE 6800-6809 BOOK

By: Ronald W. Anderson
As published in 68 MICRO JOURNAL™

The publishers of 68 MICRO JOURNAL are proud to announce the publication of Ron Anderson's **FLEX USER NOTES,** in book form. This popular monthly column has been a regular feature in 68 MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. Now all his columns are being published, in whole, as the most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

As a SPECIAL BONUS all the source listing in the book will be available on disk for the low price of: FLEX™ format only —5" $12.95 — 8" $16.95 plus $2.50 shipping and handling, if ordered with the book. If ordered separately the price of the disks will be: 5" $17.95 — 8" $19.95 plus $2.50 shipping and handling.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC
	(And many more)

**Over 30 TEXT files included in ASM (assembler) — PASCAL — PIC (position independent code) TSC BASIC-C, etc.

NOTE: .C1, .C2, etc. = Chapter 1, Chapter 2, etc.

This will be a limited run and we cannot guarantee that supplies will last long. Order now for early delivery. Delivery is expected to start middle of June '83.

Softcover — Large Format

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

See your local S50 dealer/bookstore or order direct from:

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601

*FLEX is a trademark of Technical Systems Consultants

Universal Data Research, Inc. Introduces

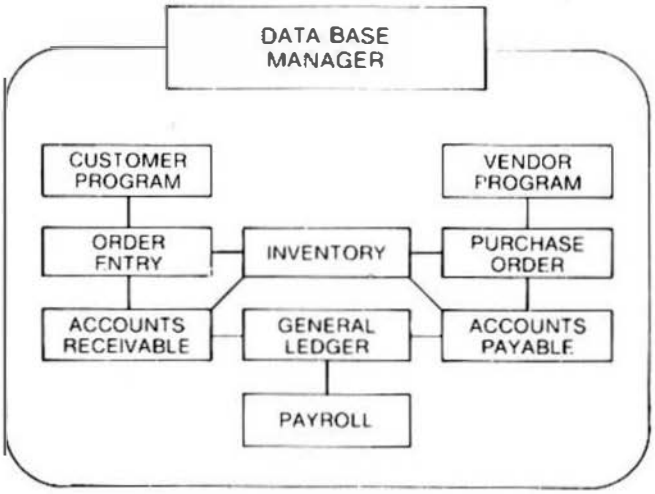
MODEMS

300 Baud Acoustic	$149.00
300 Baud Direct	$179.00
300 Baud Auto Answer	$219.00
1200 Baud Direct	$449.00
1200 Baud Auto Answer	$499.00
300/1200 Auto Answer	$549.00
300/1200 Auto Dial	$599.00
Modem 2 Port I/O Card	$119.00
with speed select	
Printer Special: Okidata 82A	$449

Dealer Inquiries Welcome ● Call or Write For Free Catalogue

New, Lower Prices

FLEX* and UniFLEX*
software for the 6809



Integrated Business Programs

	FLEX	UniFLEX
Accounts Payable	$295	$395
Accounts Receivable	$295	$395
General Ledger	$295	$395
Inventory 2	$295	$395
Payroll	$295	$395
Data Base Manager	$350	$450
Word Processing		
Software	$295	$395
WP Menu		$150
P Control		$150

UNIVERSAL DATA RESEARCH INC.

*FLEX & UniFLEX are trademarks of Technical Systems

2457 Wehrle Drive, D-68, Buffalo, NY 14221
PHONE (716) 631-3011

Dealer Inquires Welcome

master charge VISA

TMP (Total Management Planning) OS-9 SOFTWARE

THESE OFF-THE-SHELF APPLICATION PACKAGES HAVE CHANGED THE PICTURE FOR PROFESSIONAL SOFTWARE USERS

Here are the comprehensively supported OS-9 application software packages that discerning professionals are selecting today. Each one runs stand-alone or in combination with other TMP packages.

TMP/CALC is a *new-generation* "Electronic Spreadsheet" package that goes a quantum leap beyond ordinary Calc software. Superior speed, and extra features like Dynamic Calculations and Dynamic Overlay, mean you can change a cell *anywhere* on the spreadsheet and get automatic updates wherever they belong. No more "wrap-around" since output to the printer is formatted, "Help" screens prompt without erasing data, and the rows and columns on any worksheet are limited *only* by available memory.
ONLY $250

TMP/Manager is a structured database manager, and is the ultimate driving force behind any total *Management/Marketing Planning* system. Its built-in select/sort/merge module — and optimized machine code — makes it superfast, and provides unusually nimble data manipulation characteristics. Other outstanding features are detailed prompting, easy identification of data fields, and an excellent security system.
ONLY $300

TMP FreeForm fills the void between a structured DBMS and word-processing. It's an "electronic index card" package with an endless list of valuable applications such as parts cataloging, look-up inventories and listings of *any* kind. Type anything on up to 32,000 electronic "cards" of up to 9 pages each that reside in "file drawers" (databases). Up to 13 key-words can call each page.
ONLY $150

TMP/Front-End integrates any or all of the packages described above, and integrates TMP with word-processors, BASIC and other high-level languages.
NO CHARGE with TMP/Manager

TMP T.M.

TMP PACKAGES ARE EXCEPTIONALLY FRIENDLY. EACH COMES WITH MULTI-MEDIA TRAINING AND OUTSTANDING DOCUMENTATION

AUDIO CASSETTE TRAINING TAPES are provided with each TMP software package. These crisp, professionally produced training aids take a new user through "hands-on" exercises that instantly build confidence, and minimize reference to manuals.

CLEAR DOCUMENTATION is specially prepared for each TMP package. Remarkably readable because they are conversational in style and logically organized, this series of manuals *further* assures quick user comfort and productivity with TMP software. Includes sections on how to best utilize the system; integrating user-written software with TMP packages, and other bonus topics.

VIDEOTAPE PROGRAMS are available to TMP Dealers. They dramatically take the viewer into real business environments for a close look at how a TMP software package is utilized. Video programs can be personalized for specific organizations. Programs may be specified for most popular tape formats.

THE BOTTOM-LINE IS THIS: Software is only as effective as its training and documentation. TMP packages are *incomparable* in both areas.

RUN TMP IN A SMOKE SIGNAL CHIEFTAIN™ COMPUTER TO EXPERIENCE THE KIND OF SPEED AND POWER YOU CAN EASILY AFFORD *TODAY*

Endurance-Certified Chieftains are consistently among the fastest computers in CPU and I/O speeds according to the widely respected BENCHMARK REPORTS. That, combined with almost legendary reliability, makes Chieftains the logical choice in computers using OS-9 operating systems.

Configurations range from floppy-disk systems to multi-user, multi-tasking Winchester hard disk systems with tape back-up; performance approaching mainframes at prices you can live with.

TMP software packages for most OS-9 systems and Chieftain computers are offered exclusively from Smoke Signal. Inquiries from non-Smoke Signal dealers are invited. Call (213) 889-9340.

SMOKE SIGNAL
Chieftain Computers
31336 Via Colinas • Westlake Village, CA 91362
(213) 889-9340

ELEKTRA Means Quality

ELEKTRA CABINET Made of heavyweight 0.090" thick aluminum. Interior is 18-1/2" wide by 21-7/8" deep by 6-3/4" high. Heavy duty A.C. line cord. A.C. fuse holder. EMI filter. Fan with filter. Back panel has 10 cutouts for 'D' type data connectors. Front panel has key on/off power switch, 2 illuminated push button switches (Reset and NMI/Abort), and two cutouts for 5-1/4" disk drives. $2 00

Filler Plate for 5-1/4" drive opening $10.00 Fan Filler $10.00

POWER SUPPLY Highest quality linear power supply CONSERVATIVELY rated at 15a @ 8v, 3a @ 16v, 3a @ -16v. 3 primary inputs for light, latec, and heavy loading.
220v Version $200.00 110v Version $175.00

DISK REGULATOR CARD WITH CABLES Standard version for 2 floppy drives $50.00
Heavy duty version for 1 Winchester drive and 1 floppy drive $75.00

ELEKTRA UNIVERSAL MOTHERBOARD Heavyweight 0.125" thick, 18" long by 9" wide, 11 memory (50 pin) slots, 4 or 8 slots may be cut off for shortening to 14" or 10" lengths respectively, 8 I/O (30 pin) slots. Complete address decoding of I/O slots. Choice of 4, 8, or 16 addresses per I/O slot. Base address for I/O slots can be placed at 32, 64, or 128 byte increments respectively. 1" spacing between all memory and I/O slots. Extended addressing capability for both memory and I/O ports for meeting SS-50C bus specifications. On board baud rate generator with low and high ranges providing jumper selectable rates of 75 through 38,400 for each of the five baud rate lines. Slow device circuitry permitting 1MHz 30 pin disk controllers to run with 2 MHz 50 Pin CPU boards.
Bareboard w/documentation $80.00
Kit w/gold connectors $320.00 Assembled w/gold connectors $380.00
Kit w/tin connectors $240.00 Assembled w/tin connectors $300.00

ELEKTRA CHASSIS Includes cabinet, 110v power supply, power supply cables, standard disk regulator board with power cables, motherboard with gold square pin connectors, assembled and tested. $850.00

ELECTRA CPU 6/9 Use either the 6802 or 6808 (to run 6800 software) or 6809. Has provision for up to 32 716 Eproms, 1K scratchpad, MC6840 triple timer, and an optional baud rate generator providing baud rates from 110 through 38,400 baud in two user selectable ranges. The board supports DMA by either HALT or BUSREQ when a 6809 CPU is used. The board will run any of the MIKBUG™ compatible monitors in the 6802/6808 mode and S/BUG-E, HUMBUG, and MICROBUG in the 6809 mode. Versions of OS-9™ Level I are available.
Bareboard $50.00 Kit $225.00 Assembled $275.00
2K 6800 HUMBUG $40.00 2K 6809 MICROBUG $30.00 See 4K HUMBUGS next page.
Optional Baud Rate Generator $25.00

ELEKTRA DPS DUAL PORT SERIAL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 addresses per port with the B port 2 addresses higher than the A port or for 16 addresses per port with the B port 4 addresses higher than the A port. Each port is terminated at two 16 pin dip sockets, one socket configured for modem and the other for terminal or printer. RTS, CTS, DTR, DCD, and DTR are appropriately implemented. Each port has independent selection of baud rate. Each port allows the interrupt request to be independently jumpered to the IRO or FIRO/NMI bus line.
Bareboard $20.00 Kit $60.00 Assembled $80.00
Cables (two needed per board) Each $20.00

ELEKTRA DPP DUAL PORT PARALLEL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be used in either the 4 or 16 addresses per I/O slot configuration occupying the first four addresses of the I/O slot. The direction of the TTL buffers can be controlled by either on board jumper connectors or by a signal from the peripherals. The interrupt request lines for each port may be individually jumpered to either the IRO or FIRO/NMI bus line.
Bareboard $20.00 Kit $60.00 Assembled $80.00
Cables (two needed per board) Each $20.00

ELEKTRA D-5 Dual drive cabinet for 5-1/4" drives with power supply, line cord, fuse, power switch, and power cable to drives. 125.00
ELEKTRA HD-5 (Heavy duty version of D-5 package above) 150.00
ELEKTRA SND-5 (Super heavy duty. Powers 1 Winchester and 1 floppy) 175.00
5" ribbon cable for dual 5 1/4" disk drives 40.00
ELEKTRA D-8 Dual drive cabinet, power supply, ps cable for 8" drives 350.00
6' ribbon cable for dual 8" disk drives 45.00
GOLD 10 PIN CONNECTORS (Specify male w/square pins or female) 1.50
TIN 10 PIN CONNECTORS (Specify male w/square pins or female) 0.50

Prices and inventory are subject to change without advance notice.
This ad is our catalog.

WARNING AAA Chicago Computer Center does not provide repair or diagnostic service for customer assembled kits. AAA Chicago Computer Center does warranty and maintain service for our assembled boards. The customer should carefully take into consideration the small differential separating our kit and assembled prices when making his choice of purchase.

We have introduced our line of computer equipment with the purpose of offering the highest quality of components possible at affordable prices. These products are intended for OEM applications where it is the responsibility of the purchaser to integrate these components with suitable memory, disk controllers, drives, and software along with I/O terminals to form working computer systems.

AAA CHICAGO COMPUTER CENTER
120 CHESTNUT LANE • WHEELING, IL 60090
(312) 459-0450
Technical consultation available 4 P.M. to 6 P.M. most weekdays. Closed evenings and weekends.

TERMS Minimum order $20.00. Shipping and handling estimates within the Continental U.S., add 3% (MINIMUM $2.50). Illinois residents add 6% sales tax. We will refund your overestimated shipping and handling charges. Foreign shipping and handling, add 10% (MINIMUM $10.00). Foreign orders must be prepaid in U.S. dollars. Heavy foreign items will be shipped air freight collect. Please phone between 4 PM and 6 PM weekdays if questions arise regarding shipping fees. Master Charge, Visa, and American Express honored.

Our apology. We are not staffed to answer technical inquiries through the mail. Please phone for technical help during the hours indicated above. The too frequent changing of our inventory and prices makes it uneconomical to publish a catalog. Our ads are intended to serve that purpose. Prices and inventory are subject to change without advance notice.

$8/80 DISK DRIVES (30 day guarantee)

	1 head CDC	2 heads CDC	1 head MPI	flippy Sieme a	2 heads MPI	2 heads QUME
5-1/4", 40 tracks	225.00	298.00	250.00	260.00	325.00	350.00
5-1/4", 80 tracks	N/A	375.00	3 5.00	335.00	425.00	450.00

MPI Service Manual (Specify 40 or 80 track) 20.00
8", 77 tracks DS/DD Qume DT-8 $525.00 Remex (Special) 350.00

ELEKTRA
COMPUTER PRODUCTS

SMOOTH™ Software

(All of our software is copyrighted and all rights are reserved. Source is either supplied or optionally available at extra cost so that the purchaser can modify our programs for his own use. Licensing, however, is required for commercial resale.)

SUPER MODEM PROGRAM

Single character commands for simplicity.

Hardware independent (No interrupts required). A suitable modem is connected to an MC6850 (serial interface) and the control terminal is connected to an MC6850 (or MC6820 when used with the video version of GMXBUG™).

Transmit manually to distant computer.

Transmit disk files (text) of any length to distant computer.

Receive and save disk files (text) of any length on local disk system. If sending computer does not support an X-on/X-off protocol, then the received files are limited in size by the computer memory.

Tested for full duplex operation at speeds up to 9600 baud. (CRT terminal must be capable of operating at a baud rate higher than the one the modem is operated at.)

Half duplex option in case distant computer doesn't echo.

Echo option so user can simulate a time sharing system. (Super Modem Program doesn't support auto-answer but the source is provided for those individuals who wish to adapt our program to their special needs.)

Replaces CR with CR/LF (user option) for those using time sharing systems that don't transmit LF's.

Slow disk file transmit based on character verification (plus user installed timing loops if necessary) for use on time sharing systems to which disk files cannot be sent at speeds suggested by the baud rate.

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8".
Manual and disk with both source and object code $75.00

STANDARD MODEM PROGRAM

Same as Super Modem Program above but without ECHO option, CR/LF for CR option, slow disk file transmit option, nor X-on/X-off option. Reception of disk files is limited to those small enough to completely fit within the receiving buffer.

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8".
Manual with instructions, source listing, and flow chart, disk with both source and object code $45.00
Manual with instructions, source listing, and flow chart 25.00

MODEMS (By U.S. Robotics) (Auto-Dial is Hayes compatible)

	MANUAL ANSWER	AUTO ANS ER	AUTO-ANS AUTO-DIAL
300/1200 Baud, direct connect	N/A	449.00	499.00
1200 Baud (120 cps), direct connect	349.00	399.00	N/A
300 Baud (30 cps), direct connect	149.00	179.00	N/A
300 Baud (30 cps), acoustic	119.00	N/A	N/A

We will be closed from July 30th through August 21st.

THESE ADS ARE OUR CATALOG

GIMIX™ CLEARANCE SALE

	LIST	OUR PRICE		LIST	OUR PRICE
Relay driver board	538 66	250 00	6800 CPU board (used)	244 03	125 00
Filler Plate	14 92	10 00	Ba d rate gen board	88 93	65 00
Cable (Ser or Par I/O)	24 95	20 00	#48 floppy controller	198 48	150 00
#58 floppy controller	226 58	175 00	#26 control w/GMX FLEX	326 26	270 00
Double disk reg card	68 22	50 00	#68DMA con w/GMX FLEX	618 68	470 00
32K (with 16K $125.00)		175 00	56K memory board		425 00
64K memory board	478 67	450 00	Single prt ser , 1 cable	113 36	90 00
Dual prt ser , 2 cables	178 33	140 00	Dualprt par w/2 cables	138 32	110 00
Mainframe	1398 19	950 00	#29 Computer System	2498 29	2000 00
#59 Computer System	2988 59	2400 00	#49 Computer System	3246 49	2600 00
64 X 16 Video Boards	198 71	100 00	4K PPO PROM Bd and Burner	618 68	125 00
16KMem Bds , most w/cntrl reg		145 00	Reset/Abort Toggle Switch		20 00
93L422 RAM chips (2 needed for GIMIX DAT)				each	20 00

SMOOTH™ Software

ALL IN ONE
Editor - Text Processor - Mailing Labels
Mailing Lists - Use any CRT terminal and printer

Supports Editing commands such as bottom, change, delete, find, insert (single line), input (multiple lines), list, next, overlay (with curs r editing, character deletion and insertion), overstrike (for selected darker text), print, restart, set, top, underline, up, and verify.

Supports Text Processing commands such as block copy, block move, centering, margin justification (widen and narrow), paging, and tabbing.

Mailing Lists and Labels. Use the same mailing list disk file (with protected areas) for both mailing labels and repeat letters. Repeat letters are personally adressed to each person or selected persons on the mailing list.

Most Powerful File Handler found in any editor. Append one file to the end or another, or insert (merge) one file into another as designated by the line pointer. Print specified lines to your printer or to a disk file. Edit files larger than the text buffer. Does not produce output files when not desired. Delete disk files from the editor.

Printer commands. Control characters can be sent to the printer for format control either directly from the control terminal or by imbedding them in the text. The set command contains interface initialization and character output ro tines to support the SWTPC MP-C interface as well es the standard serial and parallel interfaces. Jumps are also provided to user supplied printer routines. User selects the port address (0 thru 7, A or B) thereby eliminating the need for the user to install printer software routines. Editor can be initialized for either 4 or 16 addresses per port.

Editor allows exiting to either the monitor or DOS and then reenter (WarmStart) without destroying previously prepared text in the buffer. The Restart command erases contents in the buffer without the user having to reload the Editor.

The Editor allows the user to toggle between full duplex (no echo) and half duplex (echo) as needed. It responds to com ands in both upper and lower case and can be used to create assembler source code and Basic programs as well as text.

Specify 6800 or 6809, SSB or FLEX™, 5" or 8"	75 00
Printed source listing is available for an additional	35 00
All-In-One, Write'n Spell, and Spell'n Fix package	250 00

Software by Technical Systems Consultant , Inc.	UniFLEX™ w/1 yr mnt	FLEX™
DOS (includes Editor and Assembler)	550 00	150 00
Editor or Assembler		50 00
68000 Cross Assembler on 6809	300 00	250 00
6809 Cross Assembler on 6800 or 6800 FLEX™ Utilities		100 00
Text Processor or Sort/Merge Package or 6809 FLEX™ Utilities	150 00	75 00
Extended Basic	200 00	100 00
Basic Precompiler (specify standard or extended)	150 00	50 00
Pascal	300 00	200 00
Debug Package or Diagnostic Package	75 00	75 00
6809 Relocating Assembler & Linking Loader	17500	150 00
Fortran (Requires Relocating Assembler & Linking Loader)	350 00	275 00
ortran (With Relocating Assembler & Linking Loader)	450 00	375 00
Cobol	750 00	500 00

SMOOTH and **ELEKTRA** are trademarks of AAA Chicago Computer Center
FLEX and **UniFLEX** are trademarks of Technical Systems Consultants, Inc.

6 NEW ELEKTRA PRODUCTS

ELEKTRA 30 PIN PROTOTYPING BOARD 20 00
ELEKTR 50 PIN PROTOTYPING BO RD 40 00
ENGINEER'S "FUN BOX" by ELEKTRA Computer cabinet with high quality 10 amp power supply, EMI filter, Fan. Large enough to hold the standard size SS-50 type motherboard. 225 00
64K STATIC MEMORY BOARDS with gold connectors (an available). Assembled and tested. With 56K RAM installed $299 00 With 64K RAM installed $349 00
12 and 24 MEG BYTE ELEKTR ADD-ON WINCHESTER SYSTEMS (See below for pricing)
ORDER — WRITE UP COMPUTER PROGRAM Screen oriented write up form with cursor editing, disk save and load, printer command using easily available universal printout forms. Computer program can be used to write up purchase orders packing list, account statements, plus more. Program is written in 6809 assembler language and does not require reverse video. Routines are included to support screen wrap around in both directions in case your video terminal does not. Source is provided so the customer can modify the program if he wishes. Comments are provided for adaptation to Lear Siegler and Televideo CRT terminals. The program can be easily adapted to other terminals if HOMEUP HOMEUP and ERASE and direct cursor addressing codes are known. CR, LF, and BS are the only other codes used. Available for 6809 FLEX. $100 00

SPECIALS

- Modems (Up to $100 00 off list). See our ad on the previous page
- SSB BFD Floppy DiskControllers (Version 3) Run FLEX or SSB DOS 150 00
- MICROBUG 12K, 6809, Baby HUMBUG by Peter Stark 30 00
- Cat 300 Baud Acoustic Modem (only 1 left) 75 00
- SWTPC 4K Memory $15.00, 8K Memory $40 00, MP-C $10 00, MP-LA $60 00
- SWTPC Chassis, PS, MB 125 00 *SWTPC MP-2A (Only 1 left) 150 00
- High speed tape reader $50 00 * MPI or Siemens 5" drives 125 00
- Remex 8" DS/DD (Specify) $350 00 * Ribbons for MX-80 $5 99

4K 6809 HUMBUG 75 00
4K 6800 HUMBUG (RAM needed at $A000 and $D000) 65 00
Other HUMBUG versions including video versions are available
Spell n Fix by Peter Stark 178 58
Write'n Spell by Peter Stark 75 11
Dynamite Disassembler 60 00
SUPER SLEUTH Disassembler System ($10 00 for OS-9 version) 99 00

Software by Microware Systems Corp.	Run-Time Package	Update	Source	Manual Only	Object w/Man.
OS-9™ Level One Operating System		75 00	400 00	40 00	200 00
OS-9™ Level Two Operating System		75 00	N/A	40 00	500 00
BASIC09™	100 00	75 00	N/A	25 00	200 00
OS-9™ Macro Text Editor			300 00	15 00	125 00
OS-9™ Interactive Assembler			300 00	10 00	125 00
OS-9™ Interactive Debugger (Disk version)			100 00	10 00	50 00
CIS Cobol Compiler	400 00	50 00	N/A	60 00	900 00
Pascal Compiler	100 00	100 00	N/A	40 00	400 00
Microware yearly support service ($200 00 for OS-9 Level 2)					75 00

SWTPC

6809 SWTPC FLEX™ Disk and manual (Disk only 15 00)			35 00
DC-4 Disk Controller ISS/DS, SD/DD, 5-1/4")			230 00
S32 Universal Static Memory Board (without memory chips)			124 50
SBUG-E (2716 compatible)	25 00	MP-R 2716 Eprom Programmer	114 50
MP-S2 Dual Port Serial	120 00	MP-L2 Dual Port Parallel	120 00
MP-N Calculator Board (kit)	54 95	MP-N (assembled)	92 00
MP-T Interr pt Timer II (kit)	92 00	MP-09 2MHz 6809 CPU Board	295 00

Smoke Signal Broadcasting

DCB-4A Double Density Controller Board for 5" and 8" with DOS	549 00
SSB DOS (Specify 6800 or 6809, BFD or DCB-4A, 5" or 8")	75 00
SE92/SA92-5 (6809 Edit/Assm for DOS)	69 95
SSB Monitor (Specify 6800/6809, 9600B/SE00B/SF7E8)	75 00
SSB version of FLEX™ (without Editor and Assembler)	150 00
LMB-1A Motherboard	389 00
SCB-69 6809 CPU Board	399 00
PAR-1 Dual Port Parallel Board	89 00
SER-2 Dual Port Serial Board with 2 Cables	129 00
Chief 90 64K Compurer System	2195 00

OS-9 and **BASIC09** are trademarks of Microware Systems Corp.
GIMIX and **GHOST** are registered trademarks of GIMIX Inc.

THE BEST 30 PIN FLOPPY DISK CONTROLLER THAT YOU CAN BUY! — THE ELEKTRA SUPER FC

Controls up to four 5 1/4" drives and four 8" drives for a total of eight system drives for both 6800 and 6809 systems. (FLEX software has a built in limit of a system mix of four drives). Improved programmed input/output (PIO) interface that allows single and double density 5 inch operation at 1MHz in both 6800 and 6809 systems. Allows single density 8 inch operation at 1MHz in both 6800 and 6809 systems. Allows both single and double density operation of both 5 inch and 8 inch drives at 2MHz in both 6800 and 6809 systems without slow device circuitry. Hardware and software compatible with existing SWTPC DC-1, DC-2, AND DC-3 type controllers. Controls virtually any floppy disk drive, any mix of drive size, recording density, track density, number of sides, motor control, with or without head load solenoids, including the new 2 Mbyte 5 inch drives. Analog phase locked loop data seperators for maximum reliability. Provides separate adjustments for 5 inch and 8 inch drives. Analog write pr compensation circuit with seperate adjustments for 5 inch and 8 inch drives. Designed to meet the data hold requirements of Western Digital floppy disk controller IC 's. hmidt trigger hysteresis input buffers for disk drive signals. Software write protect control. Interrupt/Data Request enable control. 12 option jumper areas to select just the right combination of hardware features to match your drives and software. Provides many separate option selections for 5 inch and 8 inch drive operation. Gold bus and drive cable connectors. Test connector for easy setup and field service. Single 5 volt power supply. 4 or 16 addresses per slot decoding. Ideal replacement for existing controllers to add double density data storage and 8 inch drive capability. Disk driver software and disk formatting utilities available to extend the capabilities of your current software. Get more for less with the ELEKTRA Super Floppy Controller.

ELEKTRA Super FC Super Floppy Controller (assembled and tesred)	$250 00
Disk with drivers and formatting utilities (Specify 6800 or 6809)	$30 00
EPROM with driver overlays	$30 00
TSC General FLEX or FLEX 9 both with Editor and Assembler	$150 00

THE BEST WINCHESTER DISK SYSTEM THAT YOU CAN BUY! — THE ELEKTRA WINCHESTER

Has automatic error detection and **CORRECTION** of up to 11 bit burst errors.

SS-50 bus, extended addressing cap bilities. DMA on board sector buffer, operates with 6800 and 6809 type processors. drivers included for FLEX 9 or OS-9. Drivers for FLEX 2 (6800) available for an additional $100 00. Price includes host interface, controller, drive(s), and cables.

12 Megabyte ELEKTRA single drive Winchester system	$2295.00
24 Megabyte ELEKTRA dual drive Winchester system	$3595.00
19 Megabyte ELEKTRA single drive Winchester system	$2995.00
38 Megabyte ELEKTRA dual drive Winchester system	$4695.00
ELEKTRA heavy duty regulator bd. (powers 1 floppy and 1 Winchester)	$75.00
ELEKTRA SHD-5 cabinet, pa. po cables (powers 1 floppy and 1 Winchester)	$175.00

AAA Chicago Computer Center See our ad on the previous page to your left for ordering instructions.

Technical Consultation available 4 p.m. to 6 p.m. most weekdays. Closed evenings and weekends.

Flex User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

Perfect Compiler Revisited

I wonder how many of you who are reading this were '68' Micro Journal subscribers in July 1980. In that issue appeared what I called more of an editorial than an article. I titled the discussion "The Perfect Compiler, An Engineer's Dream". A lot of water has flowed under the bridge since that writing. The 6809 has come along, and there are now literally a dozen useful compilers available for it. At the time of the original article, there were very few avelleble.

I thought it might be a good time to go back to that article and review what has happened since then, both in the available software, and in my needs, which I am certain, parallel the needs of many other microcomputer based instrument and control system designers. Though I have always longed for a simple application in which I could get away with an integer math compiler, only one or two have ever come my way, and they were usually part of a larger package with scientific function and floating point calculations required. I therefore still need a package with floating point capabilities. I have learned enough about scientific functions to be able to supply my own as a library package if necessary.

Time here for a brief recap of what I said in that previous article, since many of you won't have it handy for reference. First of All, I indicated that a compiler, to be useful in control applications must have easy access to absolute addresses in memory, for reading from, and writing to I/O ports. I mentioned that I could meet all my needs by programming in assembler, but that my application programs then became 60 or 80 pages of listing, which were hard to follow and time consuming to change and list.

At that time I thought that I would need String functions like those of BASIC. I had in mind, the applications in which a CRT or other alphanumeric display would be used. Since then I've found little use for String functions as such in my programs. That is largely due to the use of arrays of Characters to do string functions, and of course the print statements that allow the output of string literals and number format specifications.

I mentioned the need for branch control statements, and included the IF THEN ELSE, the FOR NEXT, the DO WHILE, and the REPEAT UNTIL structure. I should add, from experience since, that a CASE statement is very handy to have around also. The GOTO was mentioned as being more a hinderance than a help in writing a good program. Thinking of some of the early BASIC compilers that only allowed IF <condition> THEN <line number>, I mentioned the necessity of the "object" of a THEN being a statement rather then just a line number. Virtually all of the present compilers allow this, including the latest version of A/BASIC for the 6809. The original version of A/BASIC was the main reason for my comment.

One thing that makes a compiler much more useful, is the ease with which it is used, that is, the minimization of the number of steps (and time) required to get from the source text to the final binary object file. Even at that time, some of the compilers required running the compiler to produce an Assembler source file, running the assembler to produce a relocatable object module, and then running the Linking Loader to produce the final object file with the runtime package linked to it. Such multi-step compilers can result in more efficient code generation, and in fact some of the compilers include optimizer passes that look for certain sequences of code and replace them with shorter sequences.

All of the compilers that I have seen, operate on one statement of source code at a time. They have no "look ahead" feature.

```
VALUE = 3 * NUMBER;
ARRAY(7) = VALUE;
```

Such a pair of statements might result in code, such that the last code for the first statement would be, for example, STD 12,Y. Since the first thing the next statement will do is to get the contents of VALUE by an instruction such as LDD 12,Y, there is an extra instruction. The compiler doesn't assume that ACCD contains what was left in it by the previous statement, and it goes to get the contents of the variable VALUE again. An optimizer might, among other things, go look for such combinations of code, and eliminate the LDD operation. This fact that compilers make no assumptions about register contents is not all bad. It is possible to insert Assembler code between lines of compiler source code just because of that independence of statements. (The "C" standard, allows for "register variables" which have been implemented in some versions. A register variable is assigned to a processor register, and as such it may be used effectively to reduce code significantly. If this feature is used, the programmer must be careful not to use the register in inserting assembler code in the program.)

Some compilers (and some that are very efficient) have only single pass operation. They read the source text file and generate a binary object file directly.

I ruled out BASIC interpreters because of execution speed and size of memory required to run them. That observation still holds except for some "CONTROL BASIC" interpreters that do integer only, and have nice I/O port access facilities. My comments were that the perfect compiler for my stand alone applications should have BYTE (8 bit), INTEGER (16 bit), and REAL (24 bit mantissa and 8 bit signed exponent) numeric date types. In addition, I asked for a data type CHARACTER and STRING. I indicated that I preferred declaring variables of those data types to identifying the data type by "tacking on" a "$" or "%" to indicate String or Integer types.

There are now available three Pascal compilers that meet or exceed these requirements. Introl "C" in its latest version has REAL variables, as does Dugger's version 2. PL9 from Windrush also meets them. The following summarizes the capabilities:

COMPILER	BYTE	INTEGER	REAL
Lucidata Pascal	yes	yes	9 digits
OmegaSoft Pascal	no	yes	6 digits
TSC Pascal	no *	yes	16.8 digits
Introl C	yes **	yes	?? digits
Dugger C	yes ***	yes	6 digits
PL9	yes	yes	6 digits

* has special provisions for PEEK and POKE to ports
** CHAR and BYTE are interchangeable in C.
*** Dugger version 2 has short integer (BYTE) and long integer (3BYTE) arithmetic.

Lucidata, OmegaSoft and PL9 allow declaration of variables at absolute addresses.

Notice that floating point capabilities run from the 24 bit mantissa versions (6 or 7 decimal digits) to the largest 16 or 17 decimal digits. (Ten bits are approximately three decimal digits. The sign and exponent generally occupy one byte.) Introl "C" may have 24 bit integer (long integer) data as well. I don't have all the information on that product, though I have all the others.

I mentioned that I thought arrays of 2 dimensions at least, were required. I've since rethought that idea, and although multidimensional arrays are nice to have for some of my calculations, it is entirely feasible to get along with single dimensioned arrays or "vectors", though data processing applications may better use two or three dimensional arrays. Let me illustrate. Pascal has the facility of what are called "scalar" data types, which can take on a set of values declared by the programmer. Suppose we want to represent a set of mathematical vectors in both rectangular and polar coordinates. In Pascal we could do the following:

```
TYPE
   VECTORS = (X,Y,R,THETA);
     (* THIS IS AN ENUMERATED OR SCALAR TYPE);

VAR
   VECTOR : ARRAY [1..10,X..THETA] OF REAL;
```

Now we can access a vector by VECTOR [3,X] or VECTOR [7,THETA]. While that is very clean, it can be done very similarly without a two dimensional array.

```
VAR
   VECT_X : ARRAY [1..10] OF REAL;
   VECT_Y : ARRAY [1..10] OF REAL;
   VECT_R : ARRAY [1..10] OF REAL;
   VECT_THETA : ARRAY [1..10] OF REAL;
```

Now we can access the same vector quantities by VECT_X [3] or VECT_THETA [7]. The program reads almost the same, and I guarantee that single dimensioned array values are accessed faster than multi dimensioned array values. An alternative to multi-dimensional arrays in Pascal is the RECORD structure. Using that facility, the components of a vector could be accessed as VECTOR.X, VECTOR.THETA, etc.

I've programmed in Pascal extensively since writing that article, and I found that string functions like those in BASIC are totally unnecessary. Arrays of Characters are very straightforward for access and manipulation. None of the compilers mentioned here have much in the way of string functions except PL9 which has some in a library module.

The Boolean operators AND, OR, NOT, and XOR are available in all the above compilers. Those that have BYTE variables allow their use on BYTE values as well as INTEGERS.

In the area of compilers being compatible with Assembler code, Various approaches have been taken. Pascal compilers tend to allow "External Procedures". The compiler instructions indicate some protocol with regard to passing parameters to the External, either on the stack or in registers. Generally the compiler will take care of the stack operations, the user simply using the values passed to it and returning. This approach works very well, but another file is required to hold the Assembler External modules.

The implementations of "C" tend to take a simpler approach. The user simply flags the included assembler code with the statement "#asm". Includes assembler source code (using standard 6809 assembler mnemonics), and flags the end with "#endasm". The result is very clean inclusion of assembler code.

PL9 allows assembler code to be inserted between any other statements. Since it doesn't generate Assembler source code, but object code directly, it is necessary to include assembler code in object form. Rather than JMP $CD03, you use GEN $7E,$CD,$03; The keyword GEN signals that the code that follows should simply be passed on to the object file. While this sounds a little unhandy, Windrush has a companion Assembler called MACE. Mace is set up to translate Assembler source code into the form required by PL9, ie. those GEN statements. The GEN statements may be saved in a text file, and inserted in the PL9 program by means of the Co resident editor in PL9, which allows inserting the contents of a text file anywhere within the edit buffer contents.

Some of the above compilers have the data type BYTE, called "short integer" in "C". TSC and OmegaSoft Pascal and Introl "C" are the exceptions. TSC gets around this with a special set of functions. They allow access to ports with a pair of PEEK and POKE functions. The function PEEK reads a single byte but puts it into the lower 8 bits of an integer variable. PEEKW reads a double byte and puts it into an integer variable. POKE works the same way. OmegaSoft has a very well thought out EXTERNAL Assembler procedure mechanism, and the programmer will find it easiest to resort to this for communication with data I/O ports.

Lucidata Pascal and PL9 allow declaration of a byte or integer variable at an absolute address. OmegaSoft Pascal doesn't have BYTE variables, but a variable of type CHAR may be declared at the address of the port, and the built in procedure ORD used to convert the CHAR to an integer. This makes reading from and writing to a PORT equivalent to an assignment statement. To use PL9 as an example:

```
AT $E004 BYTE CONTROL,DATA;
```

This statement declares two variables, CONTROL at $E004, and DATA at $E005. To initialize the port, the following would do nicely:

```
CONTROL = 3; /* RESET */
CONTROL = 15;
```

Now to read the input from the terminal or write to it, ie., the standard I/O routines:

```
PROCEDURE GETCHAR;
  REPEAT UNTIL CONTROL. AND 1 <> 0;
ENDPROC PORT;

PROCEDURE PUTCHAR (BYTE CHAR);
  REPEAT UNTIL CONTROL. AND 2 <> 0;
  PORT = CHAR;
ENDPROC;
```

The only thing that is not straightforward here is the ENDPROC PORT; statement. ENDPROC, of course ends a procedure. PL9 is smart enough to consider what follows the keyword PROCEDURE to be a compound statement terminated with ENDPROC; The BEGIN END pair required in Pascal Procedures is not needed. ENDPROC PORT; makes the procedure the equivalent of a FUNCTION in Pascal, and returns the value of the variable PORT. In a program you would use the statement, CHAR = GETCHAR; or PUTCHAR ('A');. Now your PL9 program can talk with the terminal without SBUG-E or FLEX being present.

I mentioned several other requirements in my early article, one being memory efficiency. All of the above are reasonably memory efficient. TSC Pascal doesn't have as much flexibility with regard to selectively loading the runtime package as the others. OmegaSoft has a library function that only loads the modules that are used in the program. This reduces the object code at the expense

of more compiler steps. Lucidata has broken the runtime up into "chunks". If you don't use scientific functions or REAL arithmetic, for example, these are left out of your final runnable object file, reducing its size considerably. PL9 always loads the integer math package, adds the REAL math if you use it within the program, and allows you to select whatever other functions you need by means of selectively editing the library packages.

These compilers use the User or System stack for variables. Therefore the variables are well separated from the program code. This means that all produce rommable code. That feature is a must for my applications, and may be of no interest to others who write programs to run in the FLEX environment with a disk system present.

My early article was written before the arrival of the 6809 (at least before I had a 6809 system). The greater programming efficiency of the 6809 has done a great deal to satisfy the seemingly conflicting requirements of faster execution, smaller object code, faster compile time, and expanded features.

I am including here for your interest, a brief analysis of the compiler output of three compilers for some code that is very common in many programs. It includes the following code:

```
X=0;
Y=0;
X=X+1;
Y=Y+1;
X=X-1;
Y=Y-1;
TABLE1(X+3)=0;
TABLE2(Y+3)=0;
TABLE1(X+3)=TABLE1(X+3)+1;
TABLE2(Y+3)=TABLE2(Y+3)+1;
TABLE1(X+3)=TABLE1(X+3)-1;
TABLE2(Y+3)=TABLE2(Y+3)-1;
```

The variable X is defined as a byte (in the compilers that have BYTE data types), and TABLE1 is an array (10) of BYTES. Y and TABLE2 are defined as integers. In the compilers that don't support BYTE, I have run all as integers in order to get a comparison.

In order not to be misleading, I must explain that PL9 has been optimized for these functions. It was decided by the author that they occur so often in programs that the compiler ought to handle them very efficiently. Also notice that BYTE and INTEGER in PL9 are handled differently, again to optimize the compiler output.

OmegaSoft doesn't handle BYTE variables, so its output is only for the INTEGER calculations. PL9 arrays always start with element (0). That is, an array(10), has elements 0 through 9. Pascal allows the index to have whatever range is specified by the user. I declared the arrays [1..10], therefore Pascal must subtract 1 in order to access the first element of the array at zero offset from the base address of the array. In general, the compilers with more features and flexibility must generate a little more code. OmegaSoft has optimized the zeroing of a variable, and in fact zeros a simple integer variable with four bytes of code compared to five in PL9. OmegaSoft uses eleven bytes to add 1 to an integer while PL9 gets by with seven.

Dugger's C doesn't fare quite so well. It uses ten bytes to zero an integer, and twenty two to add 1. Adding one, involves a call to the runtime subroutine CCADDI, which is about 12 bytes. I am not trying to pick on "C", and not on this particular compiler, which is in fact rather above average in efficiency. The point is simply that some of these compilers are more fully optimized for frequently repeated operations.

After writing the above, I found a friend who has INTROL C in the integer only version. We ran the same test as I had run in Dugger's C. The only change was to eliminate the equate library inclusion, which is not necessary in INTROL C. While the compiler was happy with data type "short integer", it compiled the same code for the short type as the regular integer. It therefore is not equipped to produce more efficient code for BYTE arithmetic. It coded TABLE2(Y+3) = 0; in 19 bytes vs 20 bytes for PL9. See below for a further comparison of four of the compilers.

Looking at the compilers output for the last expression, TABLE2 (Y+3) = TABLE2 (Y+3) - 1; we find that PL9 generated 31 bytes, OmegaSoft 62, Dugger's 67, and Introl 37.

BYTE variables are not handled as efficiently as integers in Lucidata Pascal. The chief purpose of them is for accessing ports, and where RAM memory is very limited. When I ran the whole test program for Lucidata the compiler generated 300 bytes. I ran it for only the INTEGER calculations and it generated only 130 bytes. Doing the same operations with BYTES actually generated more code than doing them with integers. Knowing this, you would be likely to be sure there is plenty of RAM and only use BYTE data type in Lucidata Pascal where it is needed.

The overall byte counts are as follows:

PL9	157
Introl	261
Lucidata	300 *
OmegaSoft	376
Dugger	456 **

* Requires runtime interpreter to run.
** Requires runtime Add and Subtract routines to run.
The others are complete and "runnable".

Well, that's about what I had to say in this update of my previous thoughts about compilers. As you can see, there is a choice presently. There are several other implementations of "C" around, and several of the suppliers have promised the inclusion of REAL variables in a release to come soon.

Of the software listed here, all are single step compilers except OmegaSoft, Introl, and Dugger's. I mean, of course that they generate object code from source code with no intermediate files or steps. Introl is almost a single step compiler, in that a whole program may be compiled in a single step. It has a linker that allows combining several compiled modules into a final object module. In one of the compiler passes, an Assembler source listing file is produced, which might be handy for program debug. OmegaSoft is multi step, and it creates approximately 11 files in the process of getting from source to object code. Several of these are command files that simplify the process greatly. Dugger's is a two step compiler. The compiler produces Assembler source which is then assembled to produce the object code. The listing given here for the PL9 output was disassembled from the object code produced directly by the compiler.

Again I would like to caution you against drawing firm conclusions based on a limited scope test program. You might find rather different efficiencies for floating point calculations and still different results for a program with a great deal of terminal I/O. It would be valid to conclude from this data that the various compilers differ greatly in efficiency of code generation. Of course there is some basis for the belief that a compiler author who took the time to minimize code for some very often used cases probably wrote a compiler that is more efficient overall than one whose author didn't do that. I hasten to add that ALL these compilers are more efficient than those available two or three years ago.

Perhaps in closing, I ought to mention a couple of good integer only compilers. ABASIC is now available for the 6809 from Frank Hogg Laboratory, as is Dynasoft Pascal, a very efficient integer only version of Pascal with lots of nice features.

- - -

Rumors & Such

Received a letter from one of our advertisers last month. Seems he has been ad rtising the same products in some other computer magazines, as well as 68 MICRO JOURNAL. He has a method of determining how the customer found out about his product.

He found out exactly what many of our other advertisers found out years ago. His results from 68 MICRO JOURNAL were better than 20 times, yes, 20 times better in 68 MICRO JOURNAL than all the others he advertised in combined. No surprise to me but jolted him, especially since he had needlessly spent over $5000.00 advertising elsewhere. Seems some just never learn.

After receiving his letter and giving it some thought I decided to survey, again, a cross section of readers of 68 MICRO JOURNAL. To determine how many actually receive and read some of the other computer magazines. I got a surprise.

I knew from past surveys and other inputs that a large percentage of you did not read many or any other computer magazines. Excluding the CoCo user. What surprised me was that the number of you reading other computer magazines has declined, far more than I had thought. Since the last 68XX series was dropped from a rather large, but declining readership computer magazine, the figures have changed even more than I suspected. However, some advertisers had reported results that bear out this new data.

Having gone into this before I will not bore you with the reasons this condition exist. Other than to say that it primarily exist due to their utter disregard for the Standard S50 Bus and all it entails. That they have left the field to us helps, but I wish we had more exposure in the other magazines. It would help us all, bring in new blood and all that sort of thing. There have been other 68XX magazines start up over the past 5 years. Seems they thought we were making a killing. Well, I sure have some NEWS for them. Actually they found out the hard way. They went belly up! What they did not understand is that there are just not enough users out there to support more than one magazine. Also we have been around over five years (we were the first to completely cover the 68XX), this as well as our policies of product screening (letting you only see advertised what actually performs as advertised and is not a rip-off) has developed a loyalty from you that is difficult for them to overcome. Also most advertisers find out, after a few months, that their advertising in others does not pay off, they just pay a lot more to duplicate what they were doing all along in 68 MICRO JOURNAL.

Another fact; if we did not do all our production, including printing and binding, it would cost more than our net profits have ever been! It cost just as much practically to do a press run of 150,000 sheets as it does 15,000 sheets. The cost of press preparation and cleanup afterwards actually is the major cost, along with the waste and spoilage that occurs with any printing job. Yes, paper is expensive, but the press time is even more expensive, including collate processing, trimming and binding. So, we have a much, much higher cost per unit than if we were printing say 150,000 or even 50,000. If we did not own our printing and binding operation, well there would probably be no 68 MICRO JOURNAL.. So they try, have cash flow problems and finally just fade away. I really feel sorry for some that have failed. I know them personally and in fact, for the most part, they are fine decent folks. Just did not understand or believe what I tried to tell them. Guess they thought it somewhat like the fox guarding the hen house.

Getting back to how many of you read or subscribe to other computer magazines, I thought you might like to know, so here is what I found according to our latest input:

Note: this excludes CoCo users

Read Others	Subscribe to Others
51%	37%

A year ago these figures were higher, by about 18%. This means, at least as I see it that most of you just don't find anything in the other computer magazines that is of particular interest.

Take note advertisers!

OMW

- - -

SWTPC Distributors Meeting

This year I saw a lot of new hardware and software. That is good. The quality of everything seems to have improved to a degree that leaves me feeling that we can continue to be viable in the marketplace, however, I condition those feelings on the premise that Standard S50 Bus machines will find their place as 'specialty' computers, as well as do-anything machines.

The mass of new systems and software now available to the computer buyer is ponderous. The computer dealer who continues to 'shotgun' the market will face ever mounting odds. To be everything to everybody is simply not possible. Sure, you can do about any application on a Standard S50 Bus computer, and just as well as any other micro (or mini in most cases), but the number of other systems offering the same capabilities has grown beyond imagination. As in every rapidly expanding profession, we MUST specialize, and **SUPPORT**. SWTPC has done just that.

This past year SWTPC has installed some impressive school labs, based on both their S/09 and S+ systems. In the process they have developed methods of smooth installation and super efficient educational lab teaching.

Each lab can accommodate from 15 to 32 students, each student able to work on a different project with little if any 'throughput' slow down. The cost to the school system, considering the teacher/student time efficiency, brings the cost, per student, to far below not only the biggies like IBM, DEC, Wang, etc., but even below the cost of Apple and Radio Shack. Now that is efficiency!

These systems are running the latest versions of UniFLEX™, and I must admit that I am pleased with the improvements TSC has wrought to UniFLEX in these latest issues. Seems that most, if any, complaints were relatively mild. UniFLEX has a lot going for it. And when the 'configurable' version is released (soon I hope), the power of your 6809 will expand to operations unknown to even many minis!

More on the SWTPC educational development in a later issue, just wanted you to know that they have not been sleeping at the switch.

New & Better Stuff

Everything I saw was better. Far better! Fa t is the vendors attending, to show their new software, were for the most part as professional as any group in the microcomputer industry. And their products reflect a maturity that is by all accounts state-of-the-art.

Southwest Technical Products Corporation

SWTPC was showing their new S+ systems with a multitude of different disk systems. ESPECIALLY IMPRESSIVE was the new 256K CPU board that slips right into their X12 CRT terminal. This has additional serial and parallel ports as well as a 5 inch floppy and 5 inch winchester disk system. I saw it running FLEX™, however, I was told that a special version of UniFLEX (booted from a 5 inch disk), would be available. If the price is in the $5000-$6000 range, which I was told it might(?) be, it will be a sure winner. Finally a 'desktop' 6809 computer with all the attributes of it's cousins. I am impressed!

Also shown but not running was the SWTPC 68000/08 CPU card. From all indications it should expand the utilization of most all S50 bus computers. Rumor was that TSC is doing a special UniFLEX for the 68000/08.

SWTPC Maintenance Support

This year a presentation was made by Fisher Scientific - Education Division and Instrument Se vice Division. Fisher Scientific is ve y well established in both fields, nationwide. It seems that Fisher Scientific will be handling all of SWTPC's nationwide service as well as acting as a 'National' educational distributor. Just how this will all work remains to be seen, but it looks promising.

MUMPS

Micronetics was there with their MUMPS disk and applications system running on the SWTPC. Developed and used in the medical field, MUMPS has attributes that are related to non-medical applications. Not being a MUMPS user (but we have had it for a couple of years) I will wait and see. Micronetics seems to have a solid hookup with both SWTPC and Motorola so that broadens their field. In these day of specializing, MUMPS has created a niche that is probably secure from other systems.

Universal ata Research, Inc.

Joel Heckman of UDRI gave a presentation of the updated versions of their software. Running on the FLEX and UniFLEX systems with time tested software that has become practically 'bulletproof', UDRI is now increasing their product line. Most impressive was their bubble memory for the S30 bus with it's disk like capabilities. When their new 1 megabyte bubble becomes available will let you know. In it's present 256K configuration it sure speeds up data access, AND I MEAN SPEEDS UP!

Quest Computers

Karl Vollbrecht demonstrated the new Quest Computers 'Rental Management Package'. This as well as some of the other specialty software, I saw there, is what has been needed for a long time. It is these type applications that broaden the scope of 6809 computers. I receive calls nearly daily inquiring about some non-advertised specialty applications. Strange that so many of you have developed software that has market potential and keep it a secret.

I heard some dealers, at this meeting, moaning about how they were not selling much..and yet they don't advertise; some may be so narrow in scope that it would not justify advertising cost, however, they could make it available through S.E. Media, or some other mass software distributor. Or at least drop me a line describing the product so I can tell those who call and inquire. For some reason, usually busyitis, the word just never gets out.

Systems, Inc.

Joan Wealing of Systems, Inc. was showing their new 'Commodities Charting Package'. Running on a FLEX system and using the Epson MX80 w/Graftrax, it sure could save the investor or speculator type a lot of time. All in all a nice piece of software... look for a review soon, as we were promised a review package. Now I need an investor or speculator type to do the review; let me know soon if you qualify.

SAGE

Chris Dixon was back this year demonstrating his SAGE development software system. Not much I can tell you as I did not get a chance to really try it out, but for those who do programming, it should be well worth the money. Everyone there who purchased it last year, had nothing but fine words for SAGE. From Microprocessor Developments LTD of England, SAGE is reported to me to be a real time saver. Again we were promised a review copy so should be able to let you know a lot more later.

This is one of those 'secret' software packages I mentioned earlier. Hopefully it will not be secret much longer. But for the price I suspect that most potential purchasers will want to know, in depth, what it is and what it can do for them. That is what reviews are all about.

Metamico Library Systems, Inc.

Maurice Leatherbury was demonstrating their 'Library' software package, running on the SWTPC system. This presentation I missed, so I cannot really tell you very much, however, I am sure that if you are in need of this type software, a call to SWTPC can put you in touch. SWTPC knows about or has a lot of fine applications available.

Other GOOD Stuff

There were a couple of other presentations I missed. I regret missing even one, but time did not allow as complete a look-see as I would desire. However, there were a couple of other demonstrations that I heard were real nice specialty software offerings. Among these were:

1. Crit Wylie showing his 'Remote Point of Sale Control System', running on the SWTPC. I did see it generate some bills or invoices and it was impressive as how it interacted with the cash registers. Seemed like more real fine software.

2. Also Joe Bethancourt was scheduled to show his 'Auto Parts Package', again I missed it. I am sure that a call to SWTPC will get you in touch with Joe. Sorry I missed this one as I know of a dealer who is looking into just that kind of software.

Conclusion

I felt that this was the best SWTPC meeting I have attended to date. A lot of new, better and ready hard/software was looked at, played with and in general impressed by. It supports what I have been telling you all along, we are not getting older, just better (seems I heard that before). The future of the Standard S50 Bus and the 6809 looks rosy. I guess we will be around quite awhile!

In closing I want to thank Dan, Lucy, Fred, Jerry, George, Chris, Joe, Sue, Gary, Norm, Tom, Betty and all the other swell folks at SWTPC for making us feel not only welcome but right at home. The only thing I regret is that George Jr. did not arrive before we had to leave. Oh, well.

OMW

- - -

COLOR User Notes

Robert L. Nay
5900 Cassandra Smith Rd.
Hixson, Tn. 37343

Well, again I have to break a promise. The "How to Program" series has been postponed a little. This month, an Update Review, and am going to climb up on the old "Soapbox" for a while. So, on with the Show!

F-MATE(RS) Version 2.2

Those of you using the Data-Comp F-MATE(RS) Color Computer FLEX Conversion know that we just updated the package to a Version 2 a few months ago; now we have up and done it again. Ron Schmidt, who developed the excellent External Terminal Package for the F-MATE(RS) Conversion, has developed a method of using a Printer along with the Terminal, through a small external Hardware Adapter (we went this way to leave the Computer as close to normal as possible). We carried the Routines one step farther, and made the complete system invisible to the User. The External Terminal Routines (now a part of the F-MATE(RS) Package at NO additional cost) are built into the overall package in such a way that you can use them just exactly like you are at present without a Terminal, or use them WITH a Terminal but without a Printer and the Hardware Adapter, or use them WITH a Terminal AND the Hardware Adapter and Printer; there is NO change in procedures or software. BUT, this did require some changes to the RSTERMIO.BIN File, which is a part of the FLEX.SYS, so it was decided to update to a Version 2.2. The Terminal EXIT Routine is now included in the Display Screen Programs, so you can exit the Terminal directly to the 51 Column Display, or go to the Terminal directly out of an extended Display Screen, rather than going to the 32x16 Screen first. The Hardware Adapter (called the ONE PLUS) will be available in a month or so from Data-Comp Division; the PC Boards are being made up at the present time. The only requirement for this System is that the External Terminal MUST have an Internal Clock(which most all have); these routines measure the Baud Rate that the Terminal is set up for, and operate the Computer accordingly. Now you can use the Printer with the Terminal even if the Terminal does NOT have a "Printer Bypass Port".

A note to any receiving Updates; you MUST make up a new FLEX.SYS, and update your P.CMD, PRINT.SYS, and Extended Display Screens (which have been renamed to V51 and V32). With some inputs from Bud Pass of Computer Systems Consultants and Bill Adams of Westchester Applied Business Systems, the Extended Display Screens have been improved some more (some added Control Functions), and have been switched over to a Green Background with Black Letters, which eliminates the Color Fringing problems. In fact, we now normally leave the Color ON, as it provides a real comfortable Display.

EDITORIAL COMMENT!!!

PLEASE NOTE --- This SECTION is going to be SUBJECTIVE; I have tried in the past to keep this Column as OBJECTIVE as possible, and to treat ALL Advertisers the same (including Data Comp and South East Media). I have reported, as objectively as I could, on the FLEX Conversions that are available for the Color Computer (and, at times, not made some comments that I SHOULD have made BECAUSE of the Data-Comp ties; for that I apologize to YOU, the READER). To date, we have received, and reported on, the INITIAL Version of FHL FLEX; and have kept you fairly up to date on the Atomtronics Controller Board and FLEX System: we have not received any other Conversions or Updates for Review. I have bent over backwards in trying to st y away from discussing the Data-Comp F-MATE(RS) Package, because I don't want to appear to be "Pushing MY Product to the Exclusion of Others". Evidently, I have failed YOU in NOT "calling them as I see them", because the letters and phone calls over the past couple of months of "If I had only known" sure point out that the lack of solid information is hurting **YOU**, the **READER**! So, consider this fair warning, and close your eyes while you read this section if you don't want any "editorializing" in the Color Users Notes (and please accept my apologies).

First a little history. When the Color Computer first came out, the '68' Micro Journal staff realized that it would put the Motorola 6809 in the hands of MANY MORE Users than were running ALL of the SS-50 Bus Products put together. Of these hundreds of thousands of Color Computer Owners, some percentage would realize the potential of a Computer, and begin looking for a way to either make effective use of the Color Computer, or find another Computer that would do the job. IF these Owners were familiar with FLEX and the SS-50 Bus Systems, that would be the obvious place for them to turn; generating an expanding base for the sales of these types of Systems and Software. We discussed this potential with many of the SS-50 Bus Software Producers at that time, but could find NO ONE interested in providing a FLEX Conversion for the Color Computer. We then took it upon OURSELVES to provide this means of revitalizing the SS-50 "foundation" of users, and signed a contract with Steve Odneal to begin work on the FLEX Conversion for the Color Computer. He has done an OUTSTANDING job, as evidenced by the Flexibility and Reliability of this package, along with the exceptionally small number of bugs that have been uncovered. In fact, all Updates have been major Enhancements, rather than "bug repairs". We first developed the **F-MATE(EX)** FLEX Conversion for the EXATRON Disk Controller (months ahead of anyone else), which was the first available Disk Controller for the Color Computer. EXATRON never came through with the promised Double Density Disk Controller, so we looked for a better way.

In the summer of 1981, we (meaning Steve Odneal and Data-Comp), along with Mike Wolf of Atomtronics, developed the 64K RAM Modification (which at that time required some fairly major Circuit Changes to the Color Computer). (NOTE: these were the first COMPLETE 64K Mods that I am familiar with; others had shown up as expansion port adaptations, "piggy back" mods, etc.) The "D" Boards, the Version 1.1 BASIC ROM, and the 32K RS Mods also began showing up about that time, which made the 64K Mod a lot easier. The "working" Radio Shack Disk Controllers - remember that there was a noise and chip problem with the initial units - began showing up around Christmas of 1981, and Data-Comp shipped the first F-MATE Conversion in Jan '82 (and began **Advertising** it in Feb. '82). So despite what you might read elsewhere the Data-Comp F-MATE was the **first** version of FLEX for the Color Computer. Then and there we attempted to keep as much FLEX 'standard' as possible.

Now, back to the real thrust of these comments. If there were no such thing as a "Standard", where would you and your Computer be today? The **Radio Shack BASIC** in the Color Computer is a STANDARD; every Color Computer produced has THAT version of BASIC in it. One of the things that make the FLEX Operating System so nice is that a few "Standards" (some de-facto; some by intention) have developed which allows compatibility among ALL FLEX Based Systems. One is the "Single Sided, Single Density, 35 Track" Disk Format, which allows the transfer of Disks between ANY FLEX System that uses 5" Disks (look at the problem a CP/M Software Distributor has, with his many different Disk Formats to keep up with). Another is that the TSC **Mnemonic Assembler** has become a "Standard" within the FLEX Community. The Programs published, mostly in the '68' Micro **Journal**, are written to be assembled with the TSC **Mnemonic Assembler**. Much of the available Software for the FLEX Operating System is in SOURCE CODE, and must be **ASSEMBLED** before it can be used. If you are trying to learn the 6809's Assembly Language, you **MUST** have an Assembler that is "Tried and Proven". It is hard enough to learn when you have an Assembler that you can **TRUST**, and that puts out the same thing that is in a Listing that you are working from. There are some EXCELLENT Assemblers available that are NOT TSC's, but they are primarily supporting advanced features that the normal user does not need.

The Data-Comp **F-MATE(RS)** and **TSC General FLEX Package** provide as **STANDARD** a package as we know how to provide. This Package provides the **COMPLETE TSC Documentation Package**, including the "FLEX Adaptation Guide", the "Advanced Programmers Guide", the Complete "FLEX Users Manual", along with the "Editor Manual" and the "Assembler Manual". TSC provides NUMEROUS "Standard" entry points for the FLEX Operating System in their documentation, **IF YOU HAVE THEIR DOCUMENTATION!** No, it is not "Cheap", and you don't get it with the other FLEX Conversions. This Package of Manuals can be purchased from TSC (or Data-Comp, or FHL, or any other TSC Dealers), for $50.00. The TSC Line Editor, which is a part of the General FLEX Package, is a good LINE Oriented Editor, but IT is not a "Required" item, IF you **have a GOOD, FLEX Based, Editor** (it also can be purchased separately for $50.00). Finally, the General FLEX Package contains the TSC **Mnemonic Assembler**, which IS required for the "normal" use of the FLEX Operating System; UNLESS you have another FLEX Based Assembler that produces the **CORRECT** Code **AND** you are capable of making any required conversions to the Source Code that you will be assembling (and, you guessed it, it normally sells for $50.00). So, if you count it all up, that means that you get the FLEX Operating System from Data-Comp for **FREE!** This only applies if you buy all the items mentioned above, which most all serious users will do.

There are other considerations, also. The **F-MATE(RS) Conversion** provides **the Source Code** for many of the Specialized Utilities, so that you can either "fine tune" the Utilities to your System, or develop other capabilities with them. For example, we have just rewritten the Printer Routines to provide a special Driver for a Tape Puncher which produces a Machine Controller Tape. This would have been a Major Project if the Printer Command had been built into FLEX.SYS, like it is in some Conversions, instead of the "normal" **P.CMD** and PRINT.SYS Files with Source Code supplied. Data-Comp knew that the 32x16 Display just wasn't going to hack it, and was the first to provide the "Extended Display" Screens. The question asked most often by Color Computer Users who are considering the purchase of the FLEX System is "Can I use Radio Shack BASIC while I am running FLEX?" To my knowledge, ALL of the FLEX Conversions furnish this capability, in one way or another. But, let ME ask you Color Computer FLEX Users; "How much do YOU use Radio Shack BASIC while running FLEX?" They are TWO DIFFERENT SYSTEMS! If you want to run Radio Shack BASIC, run it. If you want to run FLEX, run IT. The ONLY time we can find a use for using them together is when you are using the FLEX DOS to develop Software for the Color Computer. But now, you don't need to be running Radio Shack; you need a method of developing the Software under FLEX, and then TRANSFERRING it to Radio Shack. Only Data-Comp provides this capability; the F-MATE(RS) Conversion provides Utilities for transferring FLEX Files to a Radio Shack Disk, or Radio Shack Disk Files to a FLEX Disk. (Yes, F-MATE(RS) does allow you to use PURE Radio Shack BASIC from FLEX; in fact, there are TWO methods of doing this, but NOT Disk BASIC.) The BEST thing about these procedures is that "EVERYTHING IS **STANDARD**"; you do not have a DIFFERENT BASIC or FLEX System, which NO ONE ELSE IN THE WORLD IS RUNNING, or, more important, SUPPORTING!

The Data-Comp F-MATE(RS) FLEX Conversions provide several Utilities for adapting your specific "configuration" to the FLEX Operating System through the use of the "STARTUP.TXT" File. These include "User Defined Keys" on the Keyboard, setting the Disk Drive "Stepping Rates", etc., along with the normal FLEX "TTYSET.CMD" for the Display. Data-Comp made these separate Utilities so they could be easily changed, and still provide the same types of functions as the FHL or GIMIX "SETUP.CMD". The Data-Comp F-MATE(RS) FLEX Conversion does NOT have a routine to tell FLEX which Drives are Single or Double Sided, or Single or Double Density; this is built into the Disk Drivers that are part of the F-MATE(RS) Conversion Package. If you try reading a Double Sided Disk in a Single Sided Drive, you will get Disk Read Errors; A Double Sided Drive is completely happy with a Single Sided Disk. In almost two years of operation with the F- TE(RS) System, I have never had a Disk damaged from putting the wrong format disk in a different Drive; FLEX reads the System Info Record, knows how the Disk is formatted, and reacts accordingly. The only time YOU, the User, need to tell

FLEX anything about the Disk Format is when you "NEWDISK" a Disk.

The total thrust of the Data-Comp F-MATE(RS) FLEX Conversion for the Color Computer has been to keep the package as "Standard" as possible, and to provide the maximum capability for the least money. $200.00 is NOT "Cheap", but it will become less expensive as you learn to use the System, and begin to use the Software that is available for the FLEX Disk Operating System. Add it up

$200.00 for Data-Comp; - or -

$99 for FLEX
PLUS $30 for a method to copy RS to FLEX (what about FLEX to RS??) and some kind of a BASIC
PLUS $100 for a "powerful" Editor/Assembler (which is NOT STANDARD) or the $100 for the TSC Editor and Assembler (which IS)
PLUS $50 for the FULL FLEX Documentation
PLUS ???? ---- You get the message!:

Finally, but maybe the MOST important, what kind of **SUPPORT** do you get if you have a question or problem, from any supplier. If you call or write Data-Comp for help, you get "solid information" from people who are USING the System (especially if it is in writing and documented), I know as I am there nearly 24 hours a day; other places, you might get some lady who reads a bunch of information off of a printed piece of paper, or a "our Programmer is not in right now, can you call back later", etc. How long does it take to get a "bug" fixed? Is the Software Supplier receptive to comments, improvements, fixes, enhancements, etc. Often, the only way to find these things out is "the hard way". This Magazine relies on YOU, the READER, to pass on this kind of information to the other READERS. No, we don't publish invalid complaints; we check them out. But, YOU **WILL** get an answer, one way or the other; and if there really IS a Problem, it WILL be published; as will "pats on the back".

In the "Cleaning up loose ends Dept"; BOTH Stylo AND Dynacalc run with EITHER the FHL OR Data-Comp FLEX Conversions; In spite of what some Advertisements are saying. I have been running Dynacalc with the F-MATE(RS) FLEX System since June '82, and Stylo since last Aug or Sept. The ONLY problem that I know of TODAY with the Data-Comp F-MATE(RS) Package is that it will NOT run with the new TSC DEBUG Program; BUT, we are "working on it" (where have you heard THAT before?).

OK, I've had MY say. Now, I offer ANY OTHER valid FLEX Conversion Producer "Equal Time" in THIS Column, subject ONLY to the stipulation that the comments must be factual. Also, if I have mis-stated anything in this "editorial", heave a big rock and I GUARANTEE that it will appear in this Column. The whole object is to get the "Straight Scoop" to the READERS of this Magazine; maybe this will help clear up some of the confusion.

--- RLN ---

NOTE: The update cost of ANY version of F-MATE(RS) is $19.95, with the original disks returned to Data-Comp.

- - -

BASIC FORUM

James S. Upton
2631 Seventeenth Ave.
Greeley, Colorado 80631

It has been considerably more that 2 years since the unbiquitous Color Computer made it's appearance on the market. Throughout these many months the little gray box has been alternately praised and damned. Sales figures have seemingly been good, but software support has been agonizingly slow in making an appearance. Irony has been compounded by contradiction. Tandy Corp. persists in treating the CoCo as if it were little more than a game machine. The few serious Tandy releases of software(including Extended Color Basic) have been so poorly documented as to preclude their usage in the manner allowed by the design.

Woah! Here I am writing a first column, and I am beginning to sound as if I were on a soap box in Washington Square. Most of you probably arrived at this column in much the way that I did. I am probably telling you nothing new. Indeed, one suspects that we share a purpose of trying to learn now to better get the CoCo to work for us. One may also suspect that most of you possess some trace of individualism that causes you to want to "do it yourself." Perhaps I should introduce this column and my intentions for it.

Don Williams is desirous of supplying information in Micro Journal for the Color Computer which will bridge the gap between the rank beginner and the full-fledged systems programmer. That will be the goal of this column. How do we plan to achieve this intent?

The manuals which are included with the Color Computer are a reasonable introduction to BASIC programming(if you can just ignore the scattered silliness). You have probably very nearly completed those two volumes, and are asking many questions. How do I use this thing in a practical way? How do I make it work for me? How do I go beyond the not-quite-usable programs in the manual? Where do I get ideas for programs? How do I begin a program once I have an idea? How do I improve or personalize existing programs? How do I modify a program to make it fit a specific set of needs. How do I increase program speed? Is there a better BASIC for me than that which comes in the CoCo's ROM? To be succinct, most of you want better control of your CoCo through improved programming skills on your own part. Over a period of time we'll address these and other similar questions.

Here are a few attitudinal ground rules on my part. I have access to an IBM 370. For some purposes I appreciate the great speed of the FORTRAN which is available there. I also have access to a DEC 11/44. The PASCAL there employed is wonderfully clear and logical. BASIC too has its virtues. While I like the other computer languages, I also like BASIC, and am not among those who expect to see it disolaced in the near future. Perhaps the greatest virtue of BASIC is its flexibility. It can be either very forgiving, or meticulous and demanding depending on how the programmer chooses to think. Programming(using any language) can be many things. It can be entertaining, useful or challenging. It can be practical, humorous or intellectually stimulating. All are fair game for this column.

Let's examine a practical challenge.

I am a teacher. Some time ago a change of classes put me in a position that demanded the production of several "hand-out" sheets for classes quickly. Although the copies could be reproduced from computerized masters with a copy machine in little time, each page needed to be done in a multi-column format. This is unfortunately a slow and cumbersome arrangement to type into an ordinary word processing program. To make things worse, the number of columns would vary from one page to another. The challenge was to quickly write a program to serve such a purpose.

What were the mechanical parameters for the program? At that time it had to run on a 16K CoCo, and would employ the capacities of an Epson MX-80 printer. Specifically, the 15 character per inch type of the Epson would allow as many as eight columns per 8.5" page. More that anything else it had to be done quickly. Employing what might be called the "brute force programming method," I wrote Example No. 1 in little more than an hour.

The program's organization is most simple. It consists of two loops, one for entry of text and the other for printing the text. Each is preceeded by a few preparatory lines. Line 10 turns on lower case. 160 turns it off. Lines 70 and 80 accept headings for the columns. Throughout the textual input simply press <ENTER> for unused columns. Line 180 sets 15 cpi on the printer. If your printer is not an Epson, check your manual for the correct code. 210 and 220 center and print the title. 230, 240, and 250 turn on italics, print the column headings, and then turn italics off. Yes, I do have the Graphtrax package in my MX-80. Lines 240, 260 and 280-350 will fill unused areas with either dashes or blanks at your choice. Example No. 2 is representative of the product. It is an example of advanced musical analysis. You do not need to understand it, just observe the visual effect.

Type in the program, save it, and try it out. Perhaps you have some immediate use for it. If not, try it in the abstract. Enter columns of resistor

color codes, animal or plant types. stamps by countries. constellations, or whatever else comes to mind. If you can only arrive at materials for six(five. four) columns leave columns one and eight blank(<ENTER>) etc.

Now we arrive at the most important part of this column. You are going to become a critic! You have in front of you a functioning program. It works. but is not an especially good example of the programming art. It does illustrate now forgiving BASIC can be when a program is needed quickly. but it has many weaknesses.

How does one function as a critic? Ask the obvious question, what is wrong here? Go over the program carefully in both operation and the BASIC code. Where can problems occur? Where is the program not smooth in operation? Are there places where the program may "crash?" Is the program efficient? Is it convenient to use? Is it flexible? Is the BASIC code easy to follow and understand? In short, you are going to critically examine each element of the program asking if it really does the best possible job.

Stop! Do not read past this paragraph until you have carried out your examination. Make notes regarding your analysis as you go. Then we can compare our thinking.

Self criticism is essential to good programming. Perhaps criticising someone else's program will make the effort easier. To be critical is not necessarily to be negative. It implies a willingness to recognize both the effective and the ineffectual. Eventually you will want to try this on your own programs.

What do I find wrong with the program? Let's examine operations first.

During the inout stage each line number appears on the screen(lines are horizontal, columns vertical). They are difficult to read. An additional PRINT statement starting line 110 would correct this. The input procedure is itself clumsy, especially when less than eight columns are used. We'll discuss possible solutions at a later time. Line 170 has no protection. If "N" is entered, or a mistake is made, the program can lock up the computer. Solution: 175 IF ZZ$ <>"Y" THEN 180 ELSE 170. Although I thought it was needed when I wrote the program. the option of filling spaces with dashes clutters up the page. I think it is useless. Although one is allowed to make extra copies, the printer must be manually advanced for each copy, a time wasting procedure. The CLS instruction in line 180 is unnecessary, and distracts the eyes badly. The eight colum format is quick to assemble using the inherent capabilites of the printer, but it is not efficient. As the number of columns is reduced much space is wasted on the page. This should be corrected. How does one make a correction if a mistake is discovered after the text is entered? Perhaps a correction or review and correction section is needed.

The worst part of the BASIC coding is the horrifying series of IF statements extending from line 280 to 350. This procedure is clumsy and slow. If you agree with me that the dash option is not a virtue, we can quickly eliminate those lines.

There is a second major problem with the BASIC itself. The variables are all abstractions. That is nice for Algebra, but the program would be much clearer if the variable naming was related to the function. The printer variable might become PR$, while the spacing variable could be SP$. Rather than A$ the first column array might become ON$ or even ONE$(TWO$ etc.). Some versions of BASIC do not allow this practice. but ECB does as long as the first two letters of the variable are unique. Clarity is a distinct virtue in programming. If one is to easily edit or modify a program it is a necessity.

There is still another problem area. Line 20 CLEARs 5000 bytes of string space. That is not optimal. Using the MX-80 "microtype" the reasonable number of characters per page is 132 x 56 = 7392. *CLEAR 7000* would be a more reasonable instruction than what is given. Similarly the array dimensions of line 30 are arbitrary. 56 would be a better figure. This is derived from the practical nueber of lines to appear on the page.

Having given some critical hints. I will now give you a challenge. Improve this program. Correct it to be more flexible, efficient. understandable. and usable. The next time we'll examine my solution and how I arrived at it.

BASIC FORUM will be a tutorial. pedagogical, hopefully helpful and practical series on BASIC programming. While Extended Color Basic will receive much of our attention, we will also examine both versions of TSC's FLEX BASIC, and Computerware's Random Basic(FLEX). We will use various peripherals, especially the printer. to demonstrate the potential of the Color Computer. While this is not a hardware column, some will doubtless want to know that my CoCo is equipped with an Atomtronics 64K memory board, and an Atomtronics disk board. I use FLEX as an operating system. I have two BASF disk drives and a CRT-80A recorder for storage. I have made some other modifications to the computer too!

As a new columnist I am interested in your responses to what appears here. The title is "BASIC FORUM," and forum is intended. If you have ideas for subjects to be discussed. questions about specific BASIC problems, suggestions, or even complaints. I will appreciate hearing from you.

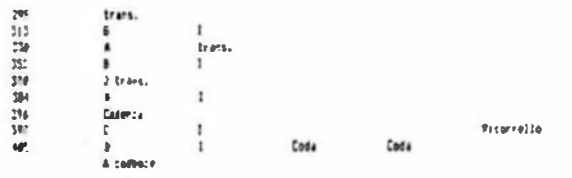
EXAMPLE NO. 1

```
10 CLS:POKE282,0
20 PRINTTAB(7)"MX-80 FORMATTER":CLEAR5000
30 DIM A$(100),B$(100),C$(100),D$(100),E$(100),F$(100),G$(100),H$(100)
40 INPUT"TITLE";AA$
50 PRINT"COLUMN HEADINGS"
60 PRINTTAB(5)STRING$(15,"-")"<-COL. LIMIT"
70 INPUT"1. ";R$:INPUT"2. ";Q$:INPUT"3. ";P$:INPUT"4. ";O$
80 INPUT"5. ";N$:INPUT"6. ";M$:INPUT"7. ";L$:INPUT"8. ";K$
90 INPUT"HOW MANY LINES";Y
100 FORX=1TOY
110 PRINTx". "
120 PRINTTAB(5)STRING$(15,"-=)"<-COL. LIMIT"
130 INPUT"1. ";A$(X):INPUT"2. ";B$(X):INPUT"3. ";C$(X):INPUT"4. ";D$(X)
140 INPUT"5. ";E$(X):INPUT"6. ";F$(X):INPUT"7. ";G$(X):INPUT"8. ";H$(X)
150 NEXT
160 POKE282,255
170 INPUT"PRINTER IS ON AND READY(Y/N)";ZZ$
180 PRINT@-2,CHR$(15):CLS
190 INPUT"DOUBLE SPACED(Y/N)";I$
200 INPUT"(1) BLANK COLUMNS OR (2) FILLED";C
210 J=LEN(AA$):K=INT((132-J)/2)
220 PRINT@-2,STRING$(K," ");AA$
230 PRINT@-2,"";CHR$(27);"4"
240 PRINT@-2,R$,Q$,P$,O$,N$,M$,L$,K$
250 PRINT@-2,"";CHR$(27);"5"
260 IFC=2THENJ$=STRING$(15,"-"):ELSE J$=STRING$(15," ")
270 FORX=1TOY
280 IF A$(X)=""THEN A$(X)=J$
290 IF B$(X)=""THEN B$(X)=J$
300 IF C$(X)=""THEN C$(X)=J$
310 IF D$(X)=""THEN D$(X)=J$
320 IF E$(X)=""THEN E$(X)=J$
330 IF F$(X)=""THEN F$(X)=J$
340 IF G$(X)=""THEN G$(X)=J$
350 IF H$(X)=""THEN H$(X)=J$
360 PRINT@-2,A$(X),B$(X),C$(X),D$(X),E$(X),F$(X),G$(X),H$(X)
370 IF I$="Y"THEN PRINT@-2,""
380 NEXT
390 INPUT"DO YOU NEED ANOTHER COPY (Y/N)";ZZ$
400 IF ZZ$="Y"THEN PRINT"POSITION THE PAPER":GOTO190
410 END
```

EXAMPLE TWO

Two (Three?) (Several?) Analyses or Structures for Mozart K.447(1)

Measure	Name	Key	Double Expo.	Single Expo.	Ritornello
1	A	I	Expo. #1	Exposition	Ritornello
9	AA	I			
29	B	I			
44	C	I			
52	D	I			
64	A	I			
68	I	mod.			
			Solo intrd. cadenza		
84	A	I	Expo. 1 2		Solo
84	AA	I			
95	trans.				
109	F	Vmin.			
122	trans.				
128	G	V			
145	A trans.				
169	H	V			
185	I trans.				
199	A	V			Ritornello
199	closing				
205	D				
223	K	Exca.	Development	Development	Solo
276	A	C	Recapitulation	Recapitulation	Ritornello
282	AA	I			



"C" User Notes

Norm Cosmo
3 Pryor Road
Natick, MA 01760

Finally, the Muse struck. More accurately, it was a matter of my schedule finally giving her time to strike. But here goes. This month it's mostly news, a couple of book reviews and some code fragments.

BOOK REVIEWS

There has been a proliferation of b ks covering the C programming language lately. Since the technical library at work was gearing up and looking for new titles, in went the order for a few of the C books that looked promising.

The books are aimed at introducing the user to the C programming language. One assumes at least some familiarity with another programming language, such as Basic. The other is written for the complete computer novice. None of them cover the entire language in complete detail, but they all hit the most salient and major features.

THE C PRIMER
Les Hancock - Morris Krieger
McGraw-Hill Book Company

This title is one of the "Byte Book" series from McGraw-Hill. The price range is $15. In the words of the authors,

> "A primer is a b k for beginners.. This primer is intended for those programmers who, while the may know something about programming, know nothing whatever about the C language; and the amount of programming knowledge we do assume our readers have is minimal."

Chapter 1 gives the history and linguistic lineage of C, what the language, is and what it isn't. I personally found this quite interesting.

Chapter 2 introduces the user to a C program that converts decimal years to Roman numerals. It shows the reader what typical C code looks like, giving the reader a quick introduction to function declarations, names in C, etc. It also gives examples of the three major format styles, which are shown below for general interest. The book and this column use style three.

```
/* style 1 */
while (i < j)
{
    i = j * j;
    j++;
}

/* style 2 */
while (i < j) {
    i = j * j;
    j++;
}

/* style 3 */
while (i < j)
    {
    i = j * j;
    j++;
    }
```

The authors then unravel the language in the following chapter order:

3 - Primary Data Types
4 - Storage Classes
5 - Operators
6 - Control Structures I
7 - Functions
8 - The C Preprocessor
9 - Arrays
10 - Pointers
11 - Control Structures II
12 - Structures
13 - Input/Output and Library Function

Also included is an Appendix that briefly touches on those areas of C not covered by the primer, on a chapter by chapter basis.

I found the book enjoyable and quick reading. The author's style is straight forward and two the point. Examples are well chosen and commented. If you are just starting out this may be just the book for you. I certainly wish it had been around when I was learning C!

Learning to Program in C
Thomas Plum
Plum Hall Inc.

This book is written for the complete newcomer to computers. It is meant to introduce the user to programming using the C language. Once more, some selected words from the author's introduction of the book:

> "This book treats the C language as a general purpose computer language for programmers concerned with portability and efficiency. No special application area is favored; examples are chosen primarily to illustrate the specific features of C".
>
> "The topic sequence has evolved over five years of teaching C language. I have tried to strike a balance between imparting the fundamentals of programming (which are independent of language and environment) and passing on useful details of the library, compilations procedures and software maintenance".
>
> "I hope the book will communicate, beyond the technical details, the delights of working in C language. C is a wonderfully designed tool for programming a variety of modern computers, elegant in its simplicity and yet powerful for many uses".

Well said Thomas Plum!! The last excerpt is also a summation of why I enjoy programming in C and writing this column; but back to the book.

The consists of nine chapters and a few appendices. Here is the Table of Contents.

0 - Introduction
1 - Computers and C
2 - Data
3 - Operators
4 - Statements and Control Flow
5 - Functions
6 - Software Development
7 - Pointers
8 - Structures
Epilogue
Appendix A
Appendix B

Each of the chapters is broken down into subchapters. The result is that once you have read the book through, you should able to quickly track down an answer using the index.

Appendix A contains a C language reference section; a list (with cross references to the book and to Unix manuals) of the "common C library; a section describing the more frequent C bugs made by programmers; and some of the common idioms often encountered in reading C programs.

I think the cost of the book is almost about justified for the bug section alone! I can guarantee you that I've done almost all of them in my o n process of learning the language, and e few (more than I'd like to admit to) still manage to sneak in.

The explanations of C idioms was interesting. It also pointed out to me how my own code segments here in the column can very easily become obscure to the neophyte.

Appendix B is an adjunct to the main text and is organized in subsections that reflect the numbered chapters. It contains the answers to questions asked in the corresponding chapter and other brief comments on the subject at hand. A nice reference in its own right.

The book is well written in an easy to understand style. There are many tables and other illustrations that succinctly wrap up the topics of a section. I particularly liked the author's use of questions (in the form of simple exercises, with the answers in the appendix) for challenging the reader to do a little work on his own.

PRODUCT PREVIEWS

I had a talk with Al Jost of Dynasoft in the latter part of April. He informed me that Dynasoft will soon becoming out with C compilers for both OS9 and FLEX. It will be a subset of full C, roughly comparable to Word's Worth's Middle C compiler.

This should be viewed as good news. I have used Dynasoft's Pascal compiler under both OS9 and FLEX and the Dynastar editor under OS9. They were pretty solid products with good documentation.

Microware will be shipping their compiler by the time you read this. I have kept in touch with Ken Kaplan at Microware on the compiler's progress. He has seemed quite satisfied with the efficiency and speed of the compiler's output.

The initial version will include all of the language except bit fields, which I haven't covered in this column. This feature should follow the initial release by a month or two.

One of the features I was really curious about was how the compiler would implement initializers. I have written a few programs that have made use of dispatch tables containing pointers to both data objects and functions. These pointers are addresses and with their initial values determined by the compiler. The problem is that OS9 can load a program any where it wants to, thereby requiring that all programs be written in a position independent fashion. How can you initialize an address that might be different each time the program is called?

Microware has the answer. It appears to be clean and shouldn't carry too much overhead in terms of time or code. It involves changing the addresses when the program is invoked, but that's all I'll say for now since I don't yet have the actual documentation.

WINDRUSH C COMPILER

I received a copy of the Windrush C compiler. Which is advertised as being the basis of the SWTP compiler. The compiler arrived on an 8 inch disk; but I have 5 inch drives. This means that I haven't actually tried the compiler, but here goes what I found in the documentation.

The compiler is made to work with TSC's new relocating assembler and linking loader. The only parts of the language not implemented are floats, doubles, bit fields and the #if directive. It does have initializers.

The documentation is terse but appears to be thorough. The features and use of the compiler are adequately explained, but don't expect a tutorial on the language. They say right up front that you won't get one and you don't.

The compiler is somewhat different than most in that it consists of an "executive" that resides in FLEX's utility area. This program actually calls in the various overlays that run in main memory and passes them the necessary parameters. In the simplest case, where there is a single program module that uses the standard library, you start the compiler and end up with a CMD executable. All the intermediate stages such as assembly and linking are handled by the executive.

For more complex cases, where there are a number of modules, you may list all the modules as either text or REL (object) modules. The executive will then do what is necessary to compile those modules that are source files.

The runtime support library is quite complete and has some rather nice functions to tie into FLEX. These include "low level" file I/O and file rename and delete functions. They have also included a function that does a quicksort! You pass the base address of an array of objects, the size of objects, the number of objects in the array and the address of a function that does a comparison of the objects. Nifty.

This looks like it has all the makings of a good compiler and I look forward to getting it back on 5 inch disk drives.

YET ANOTHER STRING FUNCTION

Code is going to be pretty sparse this month. Chances to get in and do some development have been few and far between as mentioned earlier. But here is a function that you might find handy.

Strovr() is quite simple. It takes a one string and overlays it onto another. It does not insert it, the overlay is destructive. On the other hand it does not prematurely end the first string either.

```
/*
 * overlay s2 onto s1 starting at location n.
 * If n is larger than s1, then do an append
 *
 * returns the final length of s1
 */
strovr(s1,s2,n)
    char s1[], *s2;
    int n;
    {
    int i;

    /* test n against s1 */
    if (n > (i = strlen(s1)))
        n = i;

    /* copy s2 onto s1 */
    while (*s2)
        s1[n++] = *s2++;

    /* terminate if s2 made s1 longer */
    if (n > i)
        {
        s1[n] = '\0';
        return(n);
        }
```

```
    /* else just return */
    return(i);
    }
```

The only thing here that might be construed as tricky was the initialization of I to the length of s1 within the if's test. That was done to flex your minds. It's legal and should work with all compilers or they're buggy. What happens is that I is set ot the return value of strlen(), but the value is also kept around for use in the comparison. I can't really say that it's in good style, the following is certainly clearer. On the other hand sometimes it IS cleaner if you can do something like this and C will let you.

```
I = strlen(s1);
If (n > I)
   ...
```

THAT'S A WRAP

So much for this month. Next month we will embark on a definite path. My source of code for this column is whatever I happen to be doing here on the home system, along with any nonproprietary stuff that I develop at work. I recently received a copy of the "Portable Editor in C" that was published in Dr Dobb's Journal a while back. This is a nifty screen editor written in Small C. The version I ended up with was optimized to make use of the full language and run under OS9.

I would also like it to run under FLEX, but to be really nice, the terminal I/O had better be interrupt driven. So I must come up with some scheme to get interrupt driven terminal I/O. Since the editor turns off all special character handling by the dos anyway, why not write the package right in C?
Next month we will start investigating writing various device drivers in C. It's really quite simple, even with interrupts. We'll start off with serial ports, then move onto parallel ports. I hope by then I'll also have some more news on the new compilers. Till then...

OS9 USER NOTES

By: Peter Dibble
517 Goler House
Rochester, NY 14620

A month ago I installed Basic09 on my machine. I have been proud of not having a BASIC on my computer, but OF (An OS-9/Flex copy program) requires Basic09, so I swallowed my pride and installed BASIC. I have spent too many hours breaking students of the bad habits they learned in elementary computing courses taught using BASIC to have any affection at all for that language, but I think I could learn to love Basic09. It is able to masquerade as BASIC, but it feels just like a modern structured programming language to me. I am sure that there were valid marketing reasons for including "basic" in the name of Basic09, but I wish they had named it Advanced Programming Language or something; I would feel much more comfortable learning to love the language if it had a different name.

Last Column I promised to continue wrestling with the problem of communication between processes ... Writing about processes without using technical terms is getting to be too much for me. I am going to give loose definitions of some of the important terms here.

Process or Task:
A module (Program, subroutine, or whatever) which the operating system views as an independent piece of work. A program is usually a process tho gh sometimes a program is divided into several processes.

Concurrent processes:
Strictly speaking concurrent processes must actually run at the same time. This requires a separate processor for each process. The term is sometimes loosely applied to processes (like OS-9's) that are actually using one processor in turns, but seem to be running at the same time.

Dispatch:
Give a process access to the processor. The operating system will dispatch each active process in turn. Only one process can be running at any time, so the operating system must have a way of interrupting a process as well as dispatching it.

Schedule:
Closely related to dispatch. If the operating system shows any intelligence at all about which process to dispatch next, it can be said to schedule them.

Spawn:
Create a new process. This is a more general term than FORK because not all operating systems call the operation which spawns a new process FORK.

Parent/Child:
The process that spawns a new process is called the Parent (used to be father) of the new process. The new process is said to be the child (used to be son) of the process which spawned it. The family tree analogy can be taken as far as you like; processes can have siblings, ancestors, descendants...

Asynchronous:
Not depending on the same clock.

Don't take these definitions as gospel. They are superficial -- barely enough to be useful in the context of this column.

Passing a parameter area to a FORKED process is simple, but of limited usefulness. The limitations associated with communication with processes via the parameter area are that the communication is generally one way, and that, since a copy of the parameter area is made for the new process, large parameter areas will use a lot of memory, and increase the length of time the FORK operation takes. Under OS-9 Level One, all processes share one 64K address space along with all the assorted system overhead (OS-9 itself, memory mapped I/O, etc). Spawning a new process with a 20K parameter area will cost 40K just for the parameter area (20K for the original and 20K for the new process's copy). That kind of thing can chew up a lot of memory in short order. With Level Two, the memory problem isn't so important, but, unless you have the Gimix III version of OS-9, it is time consuming to copy a large parameter area into a new address space.

Some of the characteristics of the parameter area make it possible for new families of bugs to creep into programs that use them for inter-process communications. Under OS-9 Level Two, each new process gets its own address space. There is no sign of any other process in that address space except a copy of the parameter area passed from the parent process. If the parameter area includes any addresses, they will be pointing to places that were significant in the parent's address space. In the new process's address space those addresses may be empty or contain something unexpected. The tricky

thing about this is that, under Level One, addresses in the parameter area are meaningful. Since there is only one address space, the addresses just reach out into the parent's memory and grab, or change, the data the parent pointed them at. Being able to read and change data in the parent process's memory is a mixed blessing.

Let's say you want to print the contents of an array without stopping to wait for the printer. A very good way to do this is to spawn a task to do it. If you pass the array to the new task as a parameter, everything will be fine except that, if the array is large, you may run out of memory. If you conserve memory by passing only the address of the array, everything will still be fine (under Level One) provided that neither process changes the array while the child is running. If the child changes the array, it is very likely to be a surprise for the parent. If the parent changes the array (e.g., by starting to work on new data) the child will see the changes, and print an array that is part the old one and part the new one.

It would not be too hard to track down the reason for that kind of garbled printing, but there is an especially virulent form of that bug which not only is hard to find once you set out to look for it, but also sometimes doesn't show up under most forms of testing and looks suspiciously like a hardware glitch. The operating system lets each process run for a fraction of a second, then interrupts it and dispatches another process. If you read some of another process's data, then change it and put it back (something like A = A + 1, which reads A, adds 1 to it, and stores the result in A), you can't be sure that the other process hasn't changed the data between the time you read it and the time you wrote it unless you have masked interrupts for the duration of the operation. If some process changed the value of A in the middle of the add, the new value of A will be wiped out when the result of the addition is put into A. Every process looks entirely innocent when viewed alone, but, taken together, they are chaos. If you change a program with this kind of error, even to add diagnostics, the problem may seem to disappear. The timing has to be very precise for this kind of error to show up, and (Murphy's Law being what it is) the timing is never what you want it to be. Finding and fixing this kind of bug is the kind of thing that makes a programmer want to join a commune and raise corn.

OS-9 Level Two prevents this kind of trouble with the parameter area by making addresses in the parameter area unuseable. Some programmers working in OS-9 Level One without a crystal ball to predict the nature of Level Two passed address to other processes. Their programs (I believe DYNASTAR/DYNAFORM is an example) have restrictions when they are used under OS-9 Level Two because under Level Two those addresses are not meaningful.

If addresses are included in the parameter area, and you are using Level One, a process can send data to its parent by changing the parent's variables. If you prudently don't use that questionable trick, this type of communication is like heredity: strictly from parent to child.

The parameter area is certainly the simplest path for inter-process communication, but there are are several other methods. The most powerful tool for inter-process communication is the "data module." The data module is a rather mysterious module type intended to be used to store collections of constant data. The usefulness of data modules stems from the way OS-9's LINK system service request works.

The LINK request returns the address of the module you link to. Level One simply returns the address, but Level Two must put the module in question into the address space of the process that does the LINK in order to be able to provide a meaningful address. If the module is marked "reentrant," the system memory map will be adjusted so the memory containing the module being linked to will appear in the address space of each process which is LINKed to it. This is a way to make a block of memory accessible to several processes. By making a module reentrant you assure the operating system that several processes can use the module without interfering with one another. Usually that means nobody changes the module. In the case of a shared data module it is sometimes a good idea to lie to OS-9. If you let a single process change a reentrant data module while other processes only read what's there, there is not much chance of getting into trouble. Data modules can be written into by many processes, but this requires careful management. The problems which can plague Level One users playing with two way communications through the parameter area all apply to shared data modules which are written into by more than one process.

A rather annoying problem with data modules is that they must be loaded from disk like any other module. It is possible to build a module in memory, but the system service request which forces OS-9 to include the module in its directory of modules in memory is a supervisor state request. It is possible to circumvent that restriction, but the method is too involved to tackle this month.

It is practical to have a data module with two or more "writers" because there are ways to "lock" a data module. A lock is a system for checking that a resource is free, then, if it is free, marking it "in use." Every program that uses a shared resource must check and respect the lock in order for it to be effective, but there is no way to enforce the locking in such a way that no program can get at the shared module without going through the locking protocol (GIMIX III might provide a way to do this). The easiest way to lock a module (or anything else) is to write a pair of operating system services to lock and unlock any specified resource. These services are usually called ENQ/DEQ after the sensible English words enqueue and dequeue, or P/V after two Dutch words. Dijkstra is responsible for the P/V terminology; IBM may have thought up ENQ/DEQ. Perhaps I'll write the OS-9 function handlers for P and V someday, but until those services are available, modules can be locked quite effectively in any assembly language program.

There are several instructions in the 6809 instruction set which can read and write memory all in one instruction. Altering a byte by reading and writing it in one instruction prevents any other process from accessing the byte in the middle of the alteration. The machine instructions that read and write in one instruction are: shift instructions, rotate instructions, increment, decrement, complement, and negate. The instructions which are usually used for "locking" a module are increment and decrement. The basic idea is that you set aside a locking byte in the data module with an initial value of -1. To lock the module, increment the byte, and, if increment returns with the zero flag set, continue; the module is locked. If the zero flag is not set some other process has the module locked, so decrement the locking byte, and sleep for a while... then try again. See the assembly language modules Lock, and UnLock, for examples of this procedure.

The LINK service request is only able to find modules that are already in memory. If the module

is not in memory it must be loaded from disk using the LOAD service request. This problem could be dealt with by writing two assembly language subroutines, one to do LINKs, the other to do LOADs. This offers the most flexibility, but requires the calling program to know more about OS-9 than I like. The assembly language program that accompanies this column attempts to load a module from the execution directory if it can't be found in memory. The problem with this approach is that the file which contains the data module must have the same name as the module.

The data module itself is created by the assembler. The main difference between a data module and a program module is that a data module has no permanent storage size in the module header, and no executable code . I use the execution offset field in the module header to point to the beginning of the shareable data. By convention, I use the first byte in the shareable data as a locking byte. For OS-9 Level One users, it is good to keep the module to a multiple of 256 bytes. Under Level Two, a module loaded by itself will use a multiple of the page size (usually 4096 or 2048 bytes), but a module loaded from a file containing several modules will share a page with other modules from that file if it can.

Together, the assembly language modules SLink, SUnlink, Lock, and Unlock, provide the tools necessary for a BasicO9 program to use shareable data modules. Before a data module can be used, it must be linked to; SLink returns the address at which OS-9 placed the data module. This address will be usable until the module is UnLinked. Before any data in the module is used or changed, the module should be locked by calling Lock. Lock will not return control to the calling program until it has control of the data module. It would be possible to rewrite lock so it would return with an error code if some other process had control of the data module, allowing the calling program to choose to do something other than wait if the module is not available. As soon as possible after locking the data module, it should be unlocked to release other processes waiting for the data module. Before stopping, a program that links a module should unlink it. OS-9 maintains a counter of how many times a module has been linked to, and deletes the module from memory when its link count goes to zero.

I have included two trivial BasicO9 programs to demonstrate module locking. Calc only calculates the sum of the squares of a list of numbers, but it could be the mainstay of a mail system, a matrix manipulation routine, or a print spooler (to name a few possibilities). Driver2 is a program who's greatest virtue is that it calls Calc. There are two forms of locking going on in the Driver-DataMod-Calc system: the first byte of data in DataMod is used by Lock. The second byte of data in DataMod is used for communication between Driver2 and Calc. Each process waits for this byte to take on a value set by the other process before it accesses the rest of DataMod. This is a very simple protocol which can only be used in trivial cases such as signaling between two modules. In this case, the main lock is used to prevent several modules from trying to change the communications byte at the same time. Once a process gets the lock, no other process can get it until the process holding the lock releases it. The process which has the lock can use the communications byte, and the rest of the data module, to call for the services of Calc in an organized fashion.

I use a module from last month's column called StrtTask in this set of programs. If you are especially interested in memory efficiency, merge the file containing the StrtTask module with the file containing this month's assembly language modules. Calc must be packed in order to work (at any rate, I can't puzzle out any reasonable way to use it in source form). To make the contraption go, load the file containing SLink, SUnLink, Lock, and UnLock. If StrtTask is in a separate file you might want to load that too; then start up BasicO9 and run Driver2. Driver2 will pause for a while, starting up Calc, then ask for a number five times. Give it small numbers -- they have to fit into byte variables. When all five numbers are entered, Calc will calculate the sum of their squares which will be displayed by Driver2. If you want to try it again, reply Y to the next prompt. The last thing Driver2 will do before ending is ask whether you want to shut down Calc. You do. In a system with several processes using Calc you would want to leave it running, but, with only one process using Calc, it will just be a nuisance if it is not cleaned up when its one user terminates.

Standard Terminal Support for OS-9

One of the first programs I wrote for a micro played "Life." The game starts with a given pattern, and, by repeatedly applying a set of rules, generates and displays new patterns. If the patterns are displayed properly on a CRT, the changing figures on the screen can be fascinating. (Note: Life was invented by John Horton Conway, and has been extensively discussed in Scientific American and BYTE.) I wanted my program to be usable with most terminals; so after investing a few days in the program, I spent another few weeks trying to make it "device independent." I never really finished. It was an uncommonly fast game, but, since I couldn't generalize the terminal control, no-one without a H19 will ever be able to enjoy it.

Many micros avoid this problem by not using a terminal (e.g., the Color Computer), but people, like me, who program computers without a built-in screen must either use only those control codes common to all terminals (like carriage return, and line feed), or expend a lot of effort writing special code to handle different terminals.

Full screen editors are the prime example of a type of program that must have control of some of the features of the terminal, but many other high quality programs support some of the features that most terminals share. Every program with generalized terminal support must be configured for the terminal (or terminals) it is supposed to work with as part of the installation of the program.

Some programs use a special module which contains terminal-specific code for a few crucial functions. It is simple to install a program that uses this kind of terminal control provided that the necessary module is provided. If a suitable module for your terminal is not available, a new one must be written in assembly language.

Another approach to generalized terminal control is to use a configuration program to ask questions about the terminal being used and store the information in tables which enable a single terminal control module to drive any reasonable type of terminal.

It is sad to see so much effort used solving the same problem over and over. It is so hard to write a program so it can be adjusted for use with any terminal that even some commercial programs don't do it. For small programs it can take more work to implement terminal support than to write the rest of the program. Frank Hogg Labs seems to have developed a standard for terminal control, the GOTOXY module. Once the module is installed for one program, it need not be done again except for a new

type of terminal. If every software distributor would standardize on GOTOXY, it would make life a lot easier for programmers and purchasers of software. Frank Hogg tells me the GOTOXY modules are not proprietary, so this is an alternative -- UCSD Pascal makes do with no more. Unfortunately, GOTOXY is hard to call from some languages, and supports a terribly limited set of operations.

I would like to propose a standard interface for CRT terminals. It would be much easier for Microware to build the standard control system than for me to do it, but it looks like the job is mine. I will kick the problems I find around for a month or so. Please help me with this. If I have to devise a standard in a vacuum, it may not please enough people to be widely used.

Any standard is a compromise. The most important goal is to make it easy for any programmer to use the interface. This rules out all the language-specific interfaces. The other two important goals are that all the currently existing programs with (or without) terminal control modules must continue to operate without modification, and that the interface should provide the most sophisticated terminal control possible.

Since many languages can't use GETSTAT/SETSTAT, or other exotic ways of doing I/O, I believe the standard terminal control module should either be a callable module like GOTOXY, or some form of filter. The callable module would be more efficient; but different languages call subroutines differently, and it would be sad to forsake the built-in I/O facilities of a language in order to route all terminal I/O through a single module. There are several places a module could be placed in the terminal I/O path where it could act as a filter isolating terminal specific control strings on the terminal side of the filter, and standard strings on the program side. I don't believe that the difference in efficiency between the filter and the subroutine method of terminal control is all that great. The filter method seems to be the best approach to the terminal-independent program problem.

The filter method requires that all programs act as if they are being used with some standard terminal. That terminal could be imaginary, but with so many different terminals available why invent another. Two terminals seem like attractive choices: the VT52 and the VT100. The VT100 is especially attractive because it implements the ANSII standard. It would be nice to go with the accepted standard, and I think I will finally decide to use a subset of the ANSII standard -- a subset because I don't relish the idea of trying to emulate all those flashy features on a dumb CRT. The worst disadvantage of the ANSII standard protocol is that its cursor control sequences will be hard to generate in assembly language programs. The row and column have to be in ASCII characters. It hurts me to think of a programmer being forced to include binary-to-ASCII conversion code in his program just so the terminal control module can convert the numbers back to binary. The VT52 is representative of most moderately intelligent terminals. It certainly includes every function I would want to include in the subset of the ANSII standard I plan to implement. In the short run the VT52 is a better choice than the VT100; it could be emulated more efficiently, and would be just as useful as any practical subset of the ANSII standard. Still, I believe that in the long run adhering to the most widely accepted standard is the best policy. I am looking for a good excuse to use the VT52 as the standard, but haven't found a good enough one yet.

The choice of the subset is another tricky decision. The minimum useful subset is either the direct cursor positioning command, or the set of cursor up, down, left, and right commands. Actually, home cursor is adequate for most purposes, but it takes a substantial amount of work to program for a terminal that is that dumb. There are many powerful commands that make it easier to program for a terminal, and, more important, cut down the number of characters that need to be sent to the terminal to accomplish some operation. If fewer characters need to be sent to (say) clear the screen, then the screen will clear faster and the number of interrupts the computer will need to service will be decreased. However, the more fancy terminal control commands are included in the standard, the larger the terminal control module will get.

There is no reason the filter trick can't be applied to terminal input as well as output. For some of the less powerful terminals it will be necessary to pass all input through the filter in order to know where the cursor is; however, all terminals will benefit from filtered input. An input filter will permit standard program function keys, arrow keys, the clear screen key, and perhaps some other special keys to be defined.

The following is a list of terminal control strings in descending order of likelihood to be in the subset:
Direct cursor positioning
Clear to end of line
PFkeys/Clear Key/Arrow Keys
Alternate cursor (block/underscore)/normal cursor
Highlight on/off (either reverse video or intensify)
25th (or other special) line support

The following are significantly harder:
Save cursor position/return to saved position
Insert/delete line
Delete character
Enter/leave insert character mode

I will consult everyone I can think of about this, and hope the people I don't think of will write or call me with their thoughts. After a month or two's thought, I will try to write the code to support the standard for at least one terminal. I would appreciate any help or advice I can get.

```
Microware OS-9 Assembler 2.1  05/10/83 22:43:51                    Page 001
SLink - OS-9 System Symbol Definitions

00001                    NAM  SLink
00002    ----------------------------------------------------------------
00003    *  SLink                                                        *
00004    *  Attempt to link to a module.                                 *
00005    *  If it isn't found attempt to load it.                        *
00006    *  Return the address of the module header, and the             *
00007    *  entry address.                                               *
00008    *  Errors:                                                      *
00009    *       1  Wrong number of arguments in parameter               *
00010    *          list.                                                *
00011    *    other  Return code from F$Link, or F$Load.                 *
00012    *                                                               *
00013    *  Calling sequence (from Basic09) is:                          *
00014    *    RUN Link (Module_Name, Module_Type,                        *
00015    *              Header_Addr, Entry_Addr)                         *
00016    *  Module_Name is a character string containing the             *
00017    *          name of the module which should be linked            *
00018    *          to. It should be terminated with a <CR>.             *
00019    *  Module_Type is a byte containing the language/               *
00020    *          type of the module. A data module would be           *
00021    *          0&0.                                                 *
00022    *  Header_Addr is the address of the module header of           *
00023    *          the linked module. It is returned from               *
00024    *          Link. Integer field.                                 *
00025    *  Entry_Addr is an integer field which is used to              *
00026    *          return the address of the entry point of             *
00027    *          linked module.                                       *
00028    ----------------------------------------------------------------
00029                    ifp1
```

```
00031                                   endc
00032                                   TTL   Subroutine callable from Basic09 to do Li
00033  0000 87CD0060                    MOD   LinkEnd,LinkNam,SBRTN+OBJCT,REENT+1,LinkE
00034  0009 534C696E  LinkNam           fcs   /SLink/
00035  0012 01                          fcb   1           version
00036 D 0000           LnkMemS          equ   .
00037  0013            LinkEnt
00038  0013 EC62                        ldd   2,S         get parameter count
00039  0015 10830004                    cmpd  #4          must be four
00040  0019 2633                        bne   LinkErr1
00041  001B EC8E                        ldd   14,U        get length of entry address fi
00042  001D 10830002                    cmpd  #2
00043  0021 2620                        bne   LinkErr2
00044  0023 ECE812                      ldd   18,U        get length of header address f
00045  0026 10830002                    cmpd  #2
00046  002A 2620                        bne   LinkErr2
00047  002C AE64                        ldx   4,S         Module name's address
00048  002E A6F808                      lda   [8,S]       Type/Language
00049  0031 3440                        pshs  U
00050  0033 103F00                      OS9   F$Link
00051  0036 241A                        bcc   LinkRtn     Carry clear; clean return
00052  0038 3540                        puls  U
00053  003A C1EA                        cmpb  #E$MNF      Non-existent module?
00054  003C 200E                        bra   LinkErr2    not bad error was bne
00055  003E AE64                        ldx   4,U         Module name's address
00056  0040 A6F808                      lda   [8,S]       Type/Language
00057  0043 3440                        pshs  U
00058  0045 103F01                      OS9   F$Load
00059  0048 2408                        bcc   LinkRtn
00060  004A 3540                        puls  U
00061  004C           LinkErr2
00062  004C 43                          coma
00063  004D 39                          rts               return with error code in B an
00064  004E           LinkErr1
00065  004E C4FE                        andb  #$FE        error code of 1
00066  0050 53                          comb              set carry
00067  0051 39                          rts
00068  0052           LinkRtn
00069  0052 EFF80E                      stu   [14,S]      Header address
00070  0055 10AFF812                    sty   [18,S]      data address
00071  0059 3540                        puls  U
00072  005B 1F                          clrb              clear carry
00073  005C 39                          rts               return
00074  005D C3AF0C                      EMOD
00075  0060           LinkEnd           equ   *
00076                                   NAM   SUnLink
00077                  *--------------------------------------------------*
00078                  *  SUnlink          Unlink a Linked Module.        *
00079                  *  Calling sequence (from Basic09).                *
00080                  *    RUN Unlink (Header_Addr)                      *
00081                  *  Errors:                                         *
00082                  *       1   Wrong number of arguments in parameter *
00083                  *           list.                                  *
00084                  *    other  Error code from F$Unlink.               *
00085                  *                                                   *
00086                  *  Header_Addr is the integer address of the header *
00087                  *       returned from the list request for the      *
00088                  *       module you want to unlink.                  *
00089                  *--------------------------------------------------*
00090                                   TTL   Subroutine callable from Basic09 to do Un
00091  0000 87CD002F                    MOD   UnlkEnd,UnlkNam,SBRTN+OBJCT,REENT+1,UnlkE
00092  0009 5355AE4C  UnlkNam           fcs   /SUnLink/
00093  0014 01                          fcb   1           version
00094 B 0000           ULkMemS          equ   .
00095  0015           UnlkEnt
00096  0015 EC62                        ldd   2,S         get parameter count
00097  0017 10830001                    cmpd  #1          must be one
00098  001B 2608                        bne   ULnkErr1    not one! error
00099  001D 3460                        pshs  U
00100  001F EEF806                      ldu   [6,S]       get module header's address
00101  0022 103F02                      OS9   F$Unlink    unlink the module
00102  0025 3540                        puls  U           recover U
00103                  * return code and carry set by F$Unlink
00104  0027 39                          rts               return
00105  0028           ULnkErr1
00106  0028 C4FE                        ldb   #$FE
00107  002A 53                          comb
00108  002B 39                          rts
00109  002C C97318                      EMOD
00110  002F           UnlkEnd           equ   *
00111                                   NAM   Lock
00112                  *--------------------------------------------------*
00113                  *  Lock          Lock protocol                     *
00114                  *  Wait for a "lock" byte to indicate unlocked, then *
00115                  *  lock the byte.                                  *
00116                  *  Calling sequence:                               *
00117                  *    RUN Unlock (Lock_Addr)                        *
00118                  *  Errors:                                         *
00119                  *    1   Wrong number of arguments in parameter list. *
00120                  *                                                   *
00121                  *  Lock_Addr is the integer address of the byte used *
00122                  *    for the locking protocol.                      *
00123                  *--------------------------------------------------*
00124                                   TTL   Subroutine callable from Basic09 to perfo
00125  0000 87CD0034                    MOD   LockEnd,LockNam,SBRTN+OBJCT,REENT+1,LockE
00126  0009 4C4F434B  LockNam           fcs   /LOCK/
00127  0011 01                          fcb   1           version
00128 D 0000           LokMemS          equ   .
00129  0012           LockEnt
00130  0012 EC62                        ldd   2,S         get parameter count
00131  0014 10830001                    cmpd  #1          must be one
00132  0018 2613                        bne   LockErr1    not one! error exit
00133  001A           LockLoop
00134  001A AEF804                      ldx   [4,S]       get address of lock byte
00135  001D 6C84                        inc   ,X          test and set it
00136  001F 2709                        beq   Locked
00137  0021 6A84                        dec   ,X          can't get it
00138  0023 8E0002                      ldx   #2          interval for brief sleep if una
00139  0026 103F0A                      OS9   F$Sleep
00140  0029 20EF                        bra   LockLoop
00141  002B           Locked
00142  002B 1F                          clrb              turn off carry
00143  002C 39                          rts               return
00144  002D           LockErr1
00145  002D C4FE                        ldb   #$FE
00146  002F 53                          comb
00147  0030 39                          rts               return
00148  0031 899198                      EMOD
00149  0034           LockEnd           equ   *
00150                                   NAM   UnLock
00151                  *--------------------------------------------------*
00152                  *  UnLock        Perform the Unlock protocol      *
00153                  *  Restore the "lock" byte to the unlocked         *
00154                  *  state.                                          *
00155                  *  Calling sequence:                               *
00156                  *    RUN UnLock (Lock_Addr)                        *
00157                  *  Lock_Addr is the integer address of the byte    *
00158                  *  used for the locking protocol.                  *
00159                  *--------------------------------------------------*
00160                                   TTL   Subroutine callable from Basic09 to perfo
00161  0000 87C0002A                    MOD   ULokEnd,ULokNam,SBRTN+OBJCT,REENT+1,ULokE
00162  0009 554E4C4F  ULokNam           fcs   /UnLock/
00163  0013 01                          fcb   1           version
00164 D 0000           ULokMemS         equ   .
00165  0014           ULokEnt
00166  0014 EC62                        ldd   2,S         get parameter count
00167  0016 10830001                    cmpd  #1          must be one
00168  001A 2607                        bne   ULokErr1    not one! error exit
00169  001C AEF804                      ldx   [4,S]       get address of lock byte
00170  001F 6A84                        dec   ,X          release the lock
00171  0021 1F                          clrb              set carry bit off
00172  0022 39                          rts               return
00173  0023           ULokErr1
00174  0023 C4FE                        ldb   #$FE        error code of 1
00175  0025 53                          comb              set carry
00176  0026 39                          rts
00177  0027 63C804                      EMOD
00178  002A           ULokEnd           equ   *
00179                                   NAM   DataMod
00180                                   TTL   A Lockable data module
00181                  *--------------------------------------------------*
00182                  *  This a generic data module.                     *
00183                  *  It contains a locking byte and up to 252        *
00184                  *  bytes of unspecified data.                      *
00185                  *--------------------------------------------------*
00186  0000 87CD0101                    MOD   ModEnd,ModNam,DATA,REENT+1,LockByte,0
00187  0009 44617461  ModNam           fcs   /DataMod/
00188  0014 01                          fcb   1           edition
00189  0015 FF        LockByte          fcb   -1
00190  0016 31323334  UnSpec           fcc   /1234567890123456789012345678901234567890
00191  003E 31323334                    fcc   /1234567890123456789012345678901234567890
00192  0066 31323334                    fcc   /1234567890123456789012345678901234567890
```

```
00193   008E 31323334           fcc   /12345678901234567890123456789012345678901234567890
00194   0086 31323334           fcc   /12345678901234567890123456789012345678901234567890
00195   008E 31323334           fcc   /12345678901234567890123456789012345678901234567890123/ 232
00196   00FE C37C7B             EMOD
00197   0101            ModEnd  equ   *
00198

00000 error(s)
00000 warning(s)
$01EE 00494 program bytes generated
00000 00000 data bytes allocated
$219A 08602 bytes used for symbols

PROCEDURE Calc
0000      (*
0003      (* Calculate the sum of the squares of the numbers
0035      (* stored in DataMod.
004A      (* A process signals that it wants service by storing a
0081      (* hex 01 in the byte one off the start of data in DataMod.
00BC      (* When Calc sees a 1 in that byte, it calculates the sum of
00F9      (* the squares and puts it at 7 and 8 off the start of data in
0137      (* DataMod, then sets the status byte (1 off the start) to hex 00
0178      (* indicating that calculation is done.
019F      (*
01A2      DIM Module_Name:STRING
01A9      DIM Module_Type:BYTE
01B0      DIM Header_Addr,Data_Addr:INTEGER
01BB      DIM Status_Addr,Array_Addr,Return_Addr:INTEGER
01CA      DIM sum,i:INTEGER
01D5      (*
01D6      (* Setup
01E0      (*
01E3      Module_Type=$40
01EB      Module_Name="DataMod"+CHR$(13)
01FB      RUN SLink(Module_Name,Module_Type,Header_Addr,Data_Addr)
0216      Status_Addr=Data_Addr+1
0221      Array_Addr=Data_Addr+2
022C      Return_Addr=Data_Addr+7
0237      POKE Status_Addr,0 \(* set idle (ready for work)
0258      (*
025E      (* Wait for the status byte in DataMod to
0288      (* indicate that an operation is waiting to be done.
029C      (*
02BF      WHILE PEEK(Status_Addr)<>1 DO
02CC        SHELL "SLEEP 2"
02D7      ENDWHILE
02DD      WHILE PEEK(Status_Addr)=1 DO
02E8        sum=0
02EF        FOR i=0 TO 4
02FF          sum=sum+PEEK(Array_Addr+i)*PEEK(Array_Addr+i)
0319        NEXT i
0324        (* The calculation is done. Save the result
034F        POKE Return_Addr,sum/256
035C        POKE Return_Addr+1,MOD(sum,256)
036C        (* and indicate that the results are ready
0396        POKE Status_Addr,0
039E        WHILE PEEK(Status_Addr)=0 DO
03AB          SHELL "SLEEP 2"
03B6        ENDWHILE
03BA      ENDWHILE
03BE      POKE Status_Addr,0 \(* we're dead
03D3      RUN SUnLink(Header_Addr)
03DD      BYE
PROCEDURE Driver
0000      DIM process_No,Coop_Code,Opt_Size,Lang_Type:BYTE
0013      DIM Para_L:INTEGER
001A      DIM name:STRING
0021      DIM Paras:STRING[20]
0028      (* ------------------------------------------------------------*)
0068      (* Set up to call StrFast which will fork the named            *)
00A9      (* module, passing it the parameter string in Paras.           *)
00E7      (* ------------------------------------------------------------*)
0125      name="Basic09 "
0134      process_No=0
013B      Opt_Size=0
0142      Lang_Type=$11 \(* attributes of forked module (object code, program)
017F      Paras="DTest"+CHR$(13) \(* The paras must end with <CR> for Basic09
01BA      Para_L=LEN(Paras) \(* The length of the parameters must be correct
01F2      (*
```

```
01F5    (0 Call assembler subroutines to Fork and wait for the started
0233    (0 process
0238    (0
0240    RUN StrtTask(name,process_No,Lang_Type,Parm_L,Parms,Opt_Site
0263    RUN WaitTask(process_No,Comp_Code)
0272    (0
0275    (0 Acknowledge that everything is done
0298    (0
029E    PRINT "Forked task complete"
02B6    PRINT "Completion code for process "| process_No| " was "| Comp_Code
```

SMALL DBMS

Robert Lund
P.O. Box 806
Hillside, IL. 60162

May 23, 1983

Dear Don,

Here's an interesting little program that someone may or may not be interested in. Basically (arrg.) it is a small dedicated DBMS that was intended to handle the inventory of a small electronics distributor. In actuality it has proven to be quite useful in it's own small way for a variety of DBMS applications.

The way the program stands, it loads the entire catalog into memory, and then performs it's random accesses and searching from there. This, not only for speed, but also for the reduction of head loads and seeks.

RADCAT initially, when run, looks for a file called "Catalog" on the disk. If this file is not found, it will create a blank file by the same name. After returning to FLEX, it will rename the old catalog to .BAK, and write the new updated catalog to the disk and name it "Catalog.DAT".

As it is configured, this program will store about 350 entries, or about 6 pages of 56 line entries. This of course, is only limited by the available memory and disk capacity. The program can be expanded by lengthening the table area at the end of the program. Also, RADCAT exhibits the following features:

1.) Completely menu driven.
2.) Pagenation and form headers for each page.
3.) Automatically assigns catalog numbers for cross reference.
4.) Deletes an entry when it's quantity becomes zero.
5.) Output only requires an 80 column printer.

There is an additional feature that is not immediatly obvious, and that is the fact that even non-printable control characters can be stored in the DESCRIPTION field. This will enable one to enter, and then SEARCH for an item by selected groups If certain items are grouped together by a unique control code. (eg. all plastic parts could be grouped together by adding a CNTRL P with each plastic part in the description field, and then printed on the terminal by searching for CNTRL P). Oh well, if you figured that out the first time through, your better than I am.

By the way, you tape users may find this especially interesting, as by eliminating some of the FLEX routines, you could write the table in and out of tape, and still have a random access to all the data.

Anyway, even if this program isn't useful, some of the routines may prove applicable to other ideas. The author is also interested in any improvements that may be thought of that will enhance this program, and if you send me a formatted (FLEX) disk with other like software or utilities, I will provide the source and object of this program on disk.

```
*****************************************************************
*              RADCAT - R. Lund   30-AUG-82                     *
*                                                               *
* A RCABLE catalog program.                                     *
*                                                               *
* memory utilization example: A total of 69 ($45) bytes/entry   *
*                                                               *
*         0,I 0121  00   Quantity on hand - hex digit           *
*         1,I 0122  01   Item # (2 bytes)                       *
*             0123  21   also address of this entry             *
*         3,I 0124   *                                          *
*             ~      ~                                          *
*             *      *   Description  62 bytes                  *
*             ~      ~                                          *
*        65,I    *      price         6 bytes - ASCII          *
*        68,I    ~                                             *
*             0167  00   Start of next entry                    *
*                                                               *
* GENERAL INSTRUCTIONS :                                        *
* 1.) Requires MAINs FLEX(tm) DOS                               *
* 2.) Entering into 'price' field may require leading zeros     *
*     to right justify the catalog printout.                    *
* 3.) The required files are automatically setup, and           *
*     require no initialization.                                *
* 4.) TTYSET pause function must be off. TTYSET,PS=0            *
* 5.) Auto pagenation provided for dumb printers that           *
*     don't recognize form feed 0C.                             *
*****************************************************************

F8F1  BUS    EQU   $F8F1    FIRMWARE MONITOR (NEEDS THIS WHEN BOOT)
C000  FLEX   EQU   $C000    (49000) FOR 6800
7FFE  MEMTOP EQU   $7FFE    TOP OF MEMORY
CD03  WARMS  EQU   FLEX+$0003 FLEX WARM START
CD1E  PSTRNG EQU   FLEX+$001E PRINT STRING
CD24  PCRLF  EQU   FLEX+$0024 PRINT CR,LF
CD43  OUTADR EQU   FLEX+$0043 PRINT ADDRESS
CD15  GETCHR EQU   FLEX+$0015 GET 1 ASCII CHAR.
CD3C  OUTHEX EQU   FLEX+$003C PRINT 1 HEX BYTE, 2 PRINTS TO IT
CD18  PUTCHR EQU   FLEX+$0018 OUTPUT 1 ASCII CHAR.
CD40  FCB    EQU   FLEX+$0040 FILE CONTROL BLOCK
CD3F  ERRTBL EQU   FLEX+$003F KEYBOARD ERROR
9403  FRSCLS EQU   FLEX+$1403 GLOBAL FILE CLOSE
CD2D  GETFIL EQU   FLEX+$002D GET FILE SPECIFICATION
CC14  BUFSTR EQU   FLEX+$0C14 LINE BUFFER POINTER
D406  FMS    EQU   FLEX+$1406 FILE MANAGEMENT SYSTEM
E000  PACIA  EQU   $E000    PRINTER ACIA
E004  TACIA  EQU   $E004    TERMINAL ACIA
F81F  GETADR EQU   $F81F    GET 4 CHARACTER HEX ADDRESS
E415  ADDR   EQU   $E415    FIRMWARE TEMP. STORAGE
DFE0  PORT   EQU   $DFE0    AUX. PORT VECTOR
CC49  MAPUP  EQU   FLEX+$0C49 UPPERCASE MAP

0100                ORG   $0100
0100          NEXTOP RMB   2          NEXT ENTRY STORAGE
0102          TEMP1  RMB   2          TEMP STORAGE
0104          TEMP2  RMB   2          TEMP STORAGE
0106          TEMP3  RMB   2          TEMP. STORAGE
0108          TEMP4  RMB   2          TEMP. STORAGE
010A          TEMP5  RMB   1
010B          TEMP6  RMB   1
010C          TSTRNG RMB   15         TEST STRING
011B          LINCNT RMB   1          LINES/PAGE COUNT
011C 07       FFCNT  FCB   7          FORM FEED COUNT
011D 38       LPCNT  FCB   56         LINES/PAGE COUNT

              * SET UP TABLE --
              * Zeros the catalog area.
              ************************************
011E CE 0C1D  TABCLR LDX   #TABLE
0121 4F            CLRA
0122 A7 84    TACLR  STAA  0,X
0124 30 01         INX
0126 8C 7FFE       CPX   #MEMTOP
0129 26 F7         BNE   TACLR

              * Modify Flex 'MAPUP' location to
              * allow lower case filenames.
              ************************************
012B 86 FF         LDAA  #$FF      ENABLE
012D B7 CC49       STAA  MAPUP     LOWERCASE

              * Initialize printer acia
              ************************************
0130 86 03         LDAA  #$03      RESET
0132 B7 E000       STAA  PACIA     PRINTER PORT
```

```
0135 86  11          LDAA  #$11     8 DATA, 2 STOP BITS
0137  7  E000        STAA  PACTL

                    * Read file into memory, starting
                    * at BTABLE.
                    ********************************
013A BD  05FF       JSR   READ

                    * SET UP ENTRIES *
                    * Adds all indexes into table.
                    * These are also item numbers.
                    ********************************
013D 8E  0C10       SETUP LDX   #BTABLE
0140 A6  01                 STX   1,X
0142 C6  45         SET0  LDAB  #69        CHAR./ENTRY COUNT
0144 30  01         SET4  INX
0146 BC  7FFE             CPX   #MEMTOP    CRUSH?
0149 27  07               BEQ   FIND       YES, DONE
014B 5A                   DECB             NO, BUMP TO
014C 26  FA               BNE   SET4       NEXT ITEM
014E A7  01               STX   1,X        ENTER INDEX
0150 20  F0               BRA   SET0       DO AGAIN

                    * FIND FIRST EMPTY ENTRY
                    * Finds first empty entry and
                    * saves pointer to it in NEXTOP.
                    ********************************
0152 8E  0C10       FIND  LDX   #BTABLE
0155 A6  84               LDAA  0,X        GET 1ST ENTRY
0157 27  10               BEQ   FOUND
0159 C6  45         F1    LDAB  #69        CHAR./ENTRY COUNT
015B 30  01         F2    INX
015D BC  7FFE             CPX   #MEMTOP
0160 26  00               BNE   *+1
0162 8E  0739             LDX   #MSG2      '*** ALL ENTRIES FULL ***'
0165 BD  CB1E             JSR   PSTRNG     Print it
0168 20  09               BRA   FOUND
016A 5A             *+1   DECB             CHAR. COUNT
016B 26  EE               BNE   F2
016D A6  84               LDAA  0,X
016F 27  02               BEQ   FOUND
0171 20  E6               BRA   F1         GO FOR NEXT ENTRY
0173 BF  0100       FOUND STX   NEXTOP

                    * DISPLAY OPTIONS AND GET INPUT *
                    ********************************
0176 C6  0F         DISPLAY LDAB  #15
0178 BD  CB24       DISP  JSR   PCRLF
017B 5A                   DECB
017C 26  FA               BNE   DISP
017E 8E  0570             LDX   #MSG1      'OPTIONS LIST'
0181 BD  CB1E             JSR   PSTRNG
0184 BD  CB15             JSR   GETCHR     GET ASCII CHAR.
0187 81  31               CMPA  #$31
0189 26  03               BNE   DISI
>018B 7E  0103            JMP   ENTER
018E 81  32         DISL  CMPA  #$32
0190 26  03               BNE   DIS2
0192 7E  0260             JMP   DELETE
0195 81  33         DIS2  CMPA  #$33
0197 26  03               BNE   DIS3
0199 7E  0269             JMP   SCRTCH
019C 81  34         DIS3  CMPA  #$34
019E 26  03               BNE   DIS4
01A0 7E  0282             JMP   SEARCH
01A3 81  35         DIS4  CMPA  #$35
01A5 26  03               BNE   DIS5
01A7 7E  0371             JMP   QUIT
01AA 81  36         DIS5  CMPA  #$36
01AC 26  03               BNE   DIS6
01AE 7E  0279             JMP   PRINT
01B1 20  C3         DIS6  BRA   DISPLAY

                    * FUNCTION OPTIONS *
                    ********************
                    ********************
                    * ENTER - PUT AN ENTRY INTO THE *
                    *         CATALOG.              *
                    ********************************
                    * Check to see if catalog is full
01B3 BE  0100       ENTER LDX   NEXTOP     GET CURRENT POINTER
01B6 C6  45               LDAB  #69        ITEM CHAR. COUNT
01B8 30  01         CHECK INX
01BA BC  7FFE             CPX   #MEMTOP
01BD 27  09               BEQ   FULL       ALL ENTRIES FULL
01BF 5A                   DECB
01C0 26  F6               BNE   CHECK
01C2 20  12               BRA   NOTFUL     NOT YET, GO AHEAD
01C4 00  CB24       FULL  JSR   PCRLF
01C7 BE  0739             LDX   #MSG2      'ALL FULL'
01CA BD  CB1E             JSR   PSTRNG
01CD BD  CB24             JSR   PCRLF
01D0 BD  CB24             JSR   PCRLF
01D3 7E  0361             JMP   FIN        EXIT
```

```
                    * Load 'QUANTITY' field *
0186 BD  CB24       NOTFUL JSR  PCRLF      PRINT CR, LF
0189 BD  CB24              JSR  PCRLF
018C BE  0755              LDX  #MSG3       'QUANTITY'
018F BD  CB1E              JSR  PSTRNG      Print it
0192 BD  D3CB              JSR  BYTE        GET 2 CHARS.
0195 BE  0100              LDX  NEXTOP
0198 A7  00                STAA 0,X         SAVE QUANT.
019A 30  01                INX
019C BF  0100              STX  NEXTOP      SAVE POINTER
                    * Assign 'ITEM' number and print it *
019F BD  CB24              JSR  PCRLF
01A2 BE  0769              LDX  #MSG4       'ITEM #'
01A5 BD  CB1E              JSR  PSTRNG
01A8 BE  0100              LDX  NEXTOP      POINT TO ITEM ADDR.
01AB BD  CD45              JSR  OUTADR      PRINT ITEM #
01AE 30  01                INX              POINT TO DESCRIPTION
01B0 BF  0100              STX  NEXTOP      SAVE POINTER
                    * Load 'DESCRIPTION' field *
01B3 BD  CB20              JSR  PCRLF
01B6 BE  0770              LDX  #MSG5       'DESCRIPTION'
01B9 BD  CB1E              JSR  PSTRNG
01BC BE  0100              LDX  NEXTOP
01BF C6  3E                LDAB #62         CHAR. COUNT
01C1 BD  CB15       ENTR1  JSR  GETCHR      GET ASCII CHAR.
01C4 81  00                CMPA #$00        CR?
01C6 27  12                BEQ  ENTR2
01C8 81  00                CMPA #$00        @?
01CA 26  05                BNE  ENTR3       NO
01CC 30  1F                DEX              DE-BUMP POINTER
01CE 5C                    INCB             ADD TO CHAR. COUNTER
01CF 20  F0                BRA  ENT 1       GET ANOTHER CHAR.
01D1 A7  84         ENTR3  STAA 0,X         SAVE CHAR.
01D3 30  01                INX              BUMP TO NEXT
01D5 5A                    DECB             CHAR. COUNT
01D6 26  E9                BNE  ENTR1
01D8 20  09                BRA  ENTR3       DONE HERE
01DA 86  20         ENTR2  LDAA #$20        SPACE
01DC A7  84                STAA 0,X         FILL REST
01DE 30  01                INX              WITH SPACES
01E0 5A                    DECB
01E1 26  F7                BNE  ENTR2
01E3 BF  0100       ENTR3  STX  NEXTOP      SAVE POINTER
                    * Load 'PRICE' field *
01E6 BD  CB24              JSR  PCRLF
01E9 BE  0791              LDX  #MSG6       'PRICE'
01EC BD  CB1E              JSR  PSTRNG
01EF BE  0100              LDX  NEXTOP
01F2 C6  04                LDAB #4
01F4 BD  CB15       ENTR4  JSR  GETCHR
01F7 81  00                CMPA #$00        CR?
01F9 27  09                BEQ  ENTR5
01FB A7  84                STAA 0,X
01FD 30  01                INX
01FF 5A                    DECB
0200 26  F2                BNE  ENTR4
0202 20  09                BRA  ENTR7
0204 86  20         ENTR5  LDAA #$20        SPACE
0206 A7  84         ENTR6  STAA 0,X         FILL REST
0208 30  01                INX              WITH SPACES
020A 5A                    DECB
020B 26  F9                BNE  ENTR6
020D 7E  013B       ENTR7  JMP  RETUP       DONE

                    * DELETE - DELETE AN ITEM FROM *
                    *          THE CATALOG.        *
                    * This is accomplished by simply*
                    *  entering a zero into the     *
                    *   quantity field.             *
                    ********************************
0210 BD  D37D       DELETE JSR  BYTER       GET ITEM#
0213 BE  0745              LDX  #MSG7       'ARE YOU SURE'
0216 BD  CB1E              JSR  PSTRNG
0219 BD  CB15              JSR  GETCHR      GET RESPONSE
021C 81  59                CMPA #$59        'Y'
021E 26  06                BNE  DEL1
0220 BE  0102              LDX  TEMP1
0223 4F                    CLRA
0224 A7  84                STAA 0,X         CLEAR ENTRY
0226 7E  013B       DEL1   JMP  SETUP       DONE

                    * PRINT ENTIRE CATALOG *
                    ************************
0229 BD  03FC       PRINT  JSR  PRNTR       TOGGLE OUTPUT TO PRINTER
022C BE  0C10              LDX  #TABLE
022F BF  0100              STX  NEXTOP      INITIALIZE NEXTOP
0232 86  011B       PRNT1  LDAA LPCNT       LINES/PAGE
0235 B7  011B              STAA LNCNT       LINES/PAGE COUNT
0238 F6  011C              LDAB FFCNT       FORM FEED COUNT
023B BD  CB24       PRNT2  JSR  PCRLF       PRINT CR,LF
023E 5A                    DECB
023F 26  FA                BNE  PRNT2
0241 BD  0400              JSR  HEADER      PRINT HEADER
0244 BD  044A       PRNT3  JSR  ITEM        PRINT ONE ITEM
0247 BE  0100              LDX  NEXTOP
```

```
029A 27  07          BEQ   PRNT4    DONE IF ZERO RETURNED
029C 7A  0113        DEC   LNCNT    LINE COUNT
029F 26  F3          BNE   PRNT3    PAGE NOT DONE YET
02A1 20  0F          BRA   PRNT1    WAIT PAGE
02A3 B8  0403  PRNT4 JSR   TRMNL    TOGGLE BACK TO TERMINAL
02A6 7E  013D        JMP   SETUP    EXIT

                     * SUBTRACT QUANTITY VALUE
                     ************************
02A9 B8  C024  SUBTRX JSR  PCRLF
02AC B8  C024        JSR   PCRLF
02AF B8  037B        JSR   BYTER    GET ITEM# INTO TEMP1
02B2 B8  C024        JSR   PCRLF
02B5 BE  07F4        LDI   #MSG12   'HOW MANY'
02B8 B8  C01E        JSR   PSTRNG
02BB B8  03CB        JSR   BYTE     GET QUANT. TO SUBTR.
02BE 27  0F          BEQ   SUB3     0?, DONE
02C0 BE  0102        LDI   TEMP1    POINT TO ITEM
02C3 E6  04          LDAB  0,X      GET QUANT.
02C5 5A              DECB
02C6 27  05          BEQ   SUB2
02C8 4A              DECA
02C9 27  02          BEQ   SUB2
02CB 20  F8          BRA   SUBI     DO AGAIN
02CD E7  04  SUB2    STAB  0,X      SAVE VALUE
02CF 7E  013D  SUB3  JMP   SETUP
                     * SEARCH --
                     * Search for selected string, matches of
                     * 15 characters.
                     *
                     * Temporary memory location usage:
                     *  NEXTOP - current item pointer
                     *  TEMP1  - test string pointer
                     *  TEMP2  - table pointer
                     *  TEMP3  - character count in test string
                     *  TEMP4  - current item pointer.
                     *
                     ***************************************
                     * Fill test string 'TSTRNG' with
                     *  up to 15 characters and terminate
                     *  this string with 00.
02D2 B8  CD24  SEARCH JSR  PCRLF
02D5 BE  02DC        LDI   #MSG16   'SEARCH FOR'
02D8 B8  C01E        JSR   PSTRNG   PRINT IT
02DB BE  010C        LDI   #TSTRNG  POINT TO TEXT STRING
02DE B8  CD15  SEA1  JSR   GETCHR
02E1 81  0D          CMPA  #$0D     CR?
02E3 27  08          BEQ   SEA2
02E5 A7  04          STAA  0,X      SAVE IT
02E7 30  01          INS
02E9 8C  0110        CPX   #TSTRNG+13
02EC 27  02          BEQ   SEA2
02EE 20  EE          BRA   SEA1     DO AGAIN
02F0 4F              CLRA
02F1 A7  04  SEA2    STAA  0,X      END OF STRING
02F3 B8  CD24        JSR   PCRLF
02F6 B8  040A        JSR   HEADER
                     * At this point, X is initially incremented
                     *  to the address of the 'description' field of
                     *  the first item. This address is maintained as
                     *  the current item in TEMP4.
02F9 BE  0004        LDX   #TABLE+4
02FC 8F  0108        STX   TEMP4    SAVE POINTER
02FF A6  3E  SEA3    LDAA  #62      DESC. CHAR. COUNT
0301 B7  0106        STAA  TEMP3
0304 BE  0108        LDX   TEMP4
0307 C6  45          LDAB  #69
0309 30  01  SEA4    INS
030B 8C  7FFE        CPX   #NEXTOP
030E 26  03          BNE   SEA5     NO, NOT YET
0310 7E  0341        JMP   FIN      ALL DONE
0313 5A              DECB
0314 26  F3          BNE   SEA4
0316 8F  0108        STX   TEMP4    SAVE CURRENT ITEM

0319 8F  0104        STX   TEMP2    SAVE TABLE POINTER
031C BE  010C  SEA9  LDX   #TSTRNG  POINT TO TEST STRING
031F 8F  0102        STX   TEMP1    SAVE POINTER

                     * GET 'B' FROM TESTRNG *
0322 BE  0102  SEA6  LDX   TEMP1    GET POINTER
0325 E6  04          LDAB  0,X      GET CHAR.
0327 26  02          BNE   SEA7     DONE?
0329 20  21          BRA   MATCH    ALL ITEMS MATCHED
032B 30  01  SEA7    INS            BUMP POINTER
032D 8F  0102        STX   TEMP1    SAVE IT

                     * GET 'A' FROM TABLE *
0330 BE  0104  SEA8  LDX   TEMP2    GET TABLE POINTER
0333 A6  04          LDAA  0,X      GET CHAR.
0335 30  01          INS            BUMP POINTER
0337 8C  7FFE        CPX   #NEXTOP  DONE?
033A 27  25          BEQ   FIN      YES, EXIT
033C 7A  0106        DEC   TEMP3    CHAR. COUNT
033F 27  0E          BEQ   SEA3     DONE WITH THIS ENTRY?
0341 8F  0104        STX   TEMP2    NOPE, NOT YET

                     * Compare 'A' to 'B', if not matched,
                     *  reposition 'TSTRNG' pointer (TEMP1)
                     *   back to beginning.
0344 34  04 A1E0     CBA            CHARS. MATCH?
0348 27  38          BEQ   SEA6     YES, GET NEXT SET
034A 20  00          BRA   SEA9     NO, REPOS. POINTER

034C DE  0108  MATCH LDX   TEMP4    POINT TO ENTRY
034F 30  1F          DEX
0351 30  1F          DEX            CORRECT POINTER
0353 30  1F          DEX
0355 DF  0100        STX   NEXTOP
0358 A6  04          LDAA  0,X      OPEN ENTRY?
035A 27  A3          BEQ   SEA3     YES, NO PRINT
035C B8  0465        JSR   ITEM3    NO, PRINT IT
035F 20  9E          BRA   SEA3

0361 B8  CD24  FIN   JSR   PCRLF
0364 BE  0040        LDX   #$0040
0367 B8  CD1E        JSR   PSTRNG   PRINT IT
036A B8  C015        JSR   GETCHR   WAIT FOR KEYSTROKE
036D 4F              CLRA           DUMP IT
036E 7E  0138        JMP   SETUP    EXIT

                     * EXIT ROUTINE --
                     * Final file handler, provides file
                     *  backup and exits.
                     ***************************************
0371 B8  0576  EXIT  JSR   #KWBCUT  DELETE OLD .BAK FILE
0374 B8  0487        JSR   #RENAME  RENAMES .DAT TO .BAK
0377 B8  051E        JSR   #WRITE   #WRITE NEW .DAT
037A 7E  CD03        JMP   #WARMS   EXIT

                     * GET ONE BYTE #, AND RETURN *
                     * IT IN 'TEMP1'               *
                     *******************************
037D B8  CD24  BYTER JSR   PCRLF
0380 BE  0769        LDX   #MSG4    'ITEM#'
0383 B8  CD1E        JSR   PSTRNG
0386 B9  0388        JSR   BADDR    GET 4 HEX DIGITS
0389 DF  0102        STX   TEMP1    SAVE IT
038C 39              RTS

                     * Get hex address and return in 'X'
                     *  Same as 'BADDR' in 6800 SYSMON
                     *  This routine also must echo the entered
                     *  address.
                     ***************************************
038D B8  FD1F  BADDR JSR   GETADR   GET ADDRESS
0390 BE  E419        LDX   ADDR     RETURN IT
0393 39              RTS

                     * Take one hex byte in 'A' and print
                     *  as 2 ascii digits. Leading zeros
                     *  suppressed.
                     ***************************************
0394 5F  OUTDEC CLRB
0395 7F  0103        CLR   TEMP1
0398 7F  0103        CLR   TEMP1+1
039B 81  00  D1      CMPA  #$00
039D 27  12          BEQ   PRNT     YES, PRINT
039F 4A              DECA           DEC HEX VALUE
03A0 5C              INCB           INC LSB
03A1 F7  0103        STAB  TEMP1+1  SAVE IT
03A4 C1  0A          CMPB  #$0A     OVERFLOW?
03A6 26  F3          BNE   D1       NOT YET.
03A8 7C  0102        INC   TEMP1    YES, ADD CARRY
03AB 7F  0103        CLR   TEMP1+1  CLR LSB
03AE 5F              CLRB
03AF 20  EA          BRA   D1       DO AGAIN
03B1 B6  0102  PRNT  LDAA  TEMP1
03B4 27  0E          BEQ   PRT
03B6 8A  30          ORAA  #$30
03B8 B8  CD1B        JSR   PUTCHR
03BB B6  0103  PRS   LDAA  TEMP1+1
03BE 8A  30          ORAA  #$30     MAKE ASCII
03C0 B8  CD1B        JSR   PUTCHR   PRINT IT
03C3 39              RTS
03C4 86  20  PRT     LDAA  #$20     SPACE
03C6 B8  CD1B        JSR   PUTCHR
03C9 20  F0          BRA   PRS

                     * GET 2 CHARACTERS, CONVERT 'EM TO *
                     * 1 HEX BYTE AND RETURN IN 'A'     *
                     *************************************
03CB 7F  0104  BYTE  CLR   TEMP2    CLEAR TEMP STORAGE
03CE B8  C015        JSR   GETCHR   GET MSD
03D1 84  0F          ANDA  #$0F     STRIP ASCII
03D3 B7  0106        STAA  TEMP3
03D6 27  0B          BEQ   GET3     0, GO AROUND
03D8 CA  06  GET1    LDAB  #6
03DA 7C  0104  GET2  INC   TEMP2
03DD 5A              DECB
03DE 26  FA          BNE   GET2
03E0 46              RORA
03E1 26  F9          BNE   GET1

03E3 B8  C015  GET3  JSR   GETCHR   GET LSD
03E6 81  00          CMPA  #$00     CR?
03E8 26  04          BNE   GET4     NO
03EA B6  0106        LDAA  TEMP3
03ED 39              RTS
03EE 84  0F  GET4    ANDA  #$0F     STRIP ASCII
03F0 27  06          BEQ   GET5     ALSO 0, ALL DONE
03F2 7C  0104  GET4  INC   TEMP2
03F5 44              DECA
03F6 26  FA          BNE   GET4
03F8 B6  0104  GET5  LDAA  TEMP2
03FB 39              RTS

                     * Switch output to printer
                     ****************************
03FC BE  E000  PRNTR LDX   #PACIA   PRINTER PORT ACIA
03FF DF  DFE0        STX   PORT     SAVE IN VECTOR
0402 39              RTS

                     * Switch output to terminal
                     ****************************
0403 BE  E004  TRMNL LDX   #TACIA   TERMINAL ACIA
0406 DF  DFE0        STX   PORT     SAVE IN VECTOR
0409 39              RTS

                     * PRINT CATALOG HEADER *
                     *************************
040A B8  CD24  HEADER JSR  PCRLF
040D BE  0758        LDX   #MSG8
0410 B8  2A          BSR   HEAD0
0412 C6  2B          LDAB  #43
0414 86  20          LDAA  #$20
0416 B8  CD1B  HEAD1 JSR   PUTCHR
0419 5A              DECB
041A 26  FA          BNE   HEAD1
041C BE  07DC        LDX   #MSG9
041F B8  1B          BSR   HEAD0
0421 B8  CD24        JSR   PCRLF
0424 BE  07E2        LDX   #MSG10
0427 B8  13          BSR   HEAD0
0429 C6  3E          LDAB  #62
042B 86  2D          LDAA  #$2D
042D B8  CD1B  HEAD2 JSR   PUTCHR
0430 5A              DECB
0431 26  FA          BNE   HEAD2
0433 BE  07E8        LDX   #MSG11
0436 B8  04          BSR   HEAD0
0438 B8  CD24        JSR   PCRLF
043B 39              RTS
043C A6  04  HEAD0   LDAA  0,X
043E 81  04          CMPA  #$04
0440 27  07          BEQ   HEAD5
0442 B8  CD1B        JSR   PUTCHR
0445 30  01          INS
0447 20  F3          BRA   HEAD0
0449 39              RTS
HEAD5 RTS

                     * ITEM --
                     *  With NEXTOP pointing to item,
                     *  item is printed. On exit, NEXTOP
                     *  points to next physical item 0. If
                     *  item is open or blank, NEXTOP is incremented
                     *  to next item anyway, but nothing is printed.
                     *    If a NEXTOP crash occurs, NEXTOP is
                     *  cleared on exit.
                     ***************************************************
044A DE  0100  ITEM  LDX   #NEXTOP   POINT TO ITEM 0
044D A6  04          LDAA  0,X      GET 1ST. CHAR.
044F 26  14          BNE   ITEM3    NOT 0, GO PRINT IT
0451 C6  45          LDAB  #$45     GO TO
0453 30  01  ITEM1   INS            NEXT ENTRY
0455 8C  7FFE        CPX   #NEXTOP
0458 26  03          BNE   ITEM2
045A 7E  0490        JMP   ITEM8    EXIT
045D 5A      ITEM2   DECB
045E 26  F3          BNE   ITEM1
0460 8F  0100        STX   NEXTOP
0463 20  E5          BRA   ITEM     GET NEXT ITEM
0465 B8  0394  ITEM3 JSR   OUTDEC   PRINT QUANT.
0468 86  20          LDAA  #$20
046A B8  CD1B        JSR   PUTCHR
046D C6  20          LDAB  #$20
046F B8  CD1B        JSR   PUTCHR
                     * PRINT ITEM# *
0472 DE  0100        LDX   NEXTOP   POINT TO ITEM #000.
0475 30  01          INS
0477 B8  E043        JSR   OUTA00   PRINT ITEM#
047A 30  01          INS
047C 8C  7FFE        CPX   #NEXTOP
047F 27  0F          BEQ   ITEM8    EXIT
0481 86  20          LDAA  #$20
0483 B8  CD1B        JSR   PUTCHR
0486 86  20          LDAA  #$20
0488 B8  CD1B        JSR   PUTCHR
```

```
                    * PRINT DESC. *
0488 C6  3E         ITEM4  LDAB  #62      CHAR. COUNT
048A A6  84                LDAA  0,X      GET CHAR.
048C BD  CD1D              JSR   PUTCHR   PRINT IT
048F 08                    INX
0490 3C  7FFE              CPX   #MEMTOP
0493 27  37                BEQ   ITEM8    EXIT
0495 5A                    DECB
0496 26  F1                BNE   ITEM4
0498 86  20                LDAA  #020
049A BD  CD1D              JSR   PUTCHR
                    * PRINT PRICE *
049D C6  04                LDAB  #4       PRICE CHAR. COUNT
049F A6  04         ITEM5  LDAA  0,X      GET CHAR.
04A1 81  30                CMPA  #030     ZERO?
04A3 26  11                BNE   ITEM6    SUPRESS LEADING ZERO
04A5 86  20                LDAA  #020     PRINT SPACE INSTEAD
04A7 BD  CD1D              JSR   PUTCHR
04AA 08                    INX
04AB 5A                    DECB
04AC 26  ED                BNE   ITEM5
04AE 20  12                BRA   ITEM7
04B0 A6  04         ITEM6  LDAA  0,X
04B2 BD  CD1D              JSR   PUTCHR
04B5 08                    INX
04B6 3C  7FFE              CPX   #MEMTOP
04B9 27  08                BEQ   ITEM8
04BB 5A                    DECB
04BC 26  F1                BNE   ITEM6
04BE BD  CD2A              JSR   PCRLF
04C1 BF  0100       ITEM7  STX   NEXTOP
04C4 39                    RTS
04C5 7F  0100       ITEM8  CLR   NEXTOP
04C8 7F  0101              CLR   NEXTOP+1
04CB 39                    RTS

                    * RENAME FILE ROUTINE *
                    * Renames "Catalog.DAT"
                    * to "Catalog.BAK"
                    ************************
04CC BE  CD75       RENAME LDX   #FCB+$3   GET FILENAME
04CF BF  0102              STX   TEMP1    INTO SCRATCH BYTES
04D2 BE  0810              LDX   #MSG14
04D5 BF  0104              STX   TEMP2

04D8 BE  0104       REN1   LDX   TEMP2
04DB A6  84                LDAA  0,X
04DD 81  04                CMPA  #4       END OF NAME?
04DF 27  11                BEQ   REN2     YES
04E1 08                    INX
04E2 BF  0104              STX   TEMP2

04E5 BE  0102              LDX   TEMP1
04E8 A7  84                STAA  0,X
04EA 30  01                INX
04EC BF  0102              STX   TEMP1
04EF 20  E6                BRA   REN1

04F1 0E  0000       REN2   LDX   #MSG13
04F4 DF  CC14              STX   BFPNTR
04F6 0E  CB40              LDX   #FCB
04F9 BD  CD2B              JSR   GETFIL   GET FILE SPEC.
04FC 25  08                BCS   ERROR4
04FE 86  08                LDAA  #13      RENAME FUNCTION
0500 A7  04                STAA  0,X      PUT IN FCB
0502 BD  D406              JSR   FMS      DO FUNCTION
0505 26  01                BNE   ERROR4
0507 39                    RTS

0508 BD  CD3F       ERROR4 JSR   RPTERR   REPORT ERROR
050B BD  D403              JSR   FMSCLS   CLOSE FILE
050E 7E  CD03              JMP   WARMS    EXIT TO FLEX

                    * WRITE TO DISK ROUTINE
                    * Writes catalog memory image
                    * to the disk.
                    ************************
0511 0E  0C10       WRITE  LDX   #TABLE
0514 DF  0102              STX   TEMP1    SAVE POINTER
0516 0E  0000              LDX   #MSG13   FILE NAME
0519 DF  CC14              STX   BFPNTR
051B 0E  CB40              LDX   #FCB
051E BD  CD2B              JSR   GETFIL   GET FILE NAME
0521 25  39                BCS   WRITE3   ERRORS?
0523 86  02                LDAA  #2       OPEN FOR WRITE CODE
0525 A7  04                STAA  0,X      SERIAL WRITE
0527 BD  D406              JSR   FMS      CALL FMS
052A 26  30                BNE   WRITE3   ERRORS?
052C 6F  04                CLR   0,X      SERIAL WRITE
052E 86  FF                LDAA  #$FF
0530 A7  0B 33             STAA  $B,X     NO COMPRESSION
                    * WRITE MEMORY IMAGE TO DISK
0533 0E  0102       WRITE1 LDX   TEMP1    GET TABLE POINTER
0536 A6  00                LDAA  0,X      GET CHAR. IN TABLE
0538 0E  CB40              LDX   #FCB     POINT TO FCB
053B BD  D406              JSR   FMS      CALL FMS
053E 26  1C                BNE   WRITE3   ERRORS?
0540 BE  0102              LDX   TEMP1    RESTORE POINTER
0543 08                    INX            BUMP IT
0544 BC  7FFE              CPX   #MEMTOP  DONE?
0547 27  05                BEQ   WRITE2   ALL DONE
0549 BF  0102              STX   TEMP1    NO. SAVE POINTER
054C 20  E4                BRA   WRITE1   DO NEXT CHAR.
                    * CLOSE FILE AND EXIT
054E 0E  CB40       WRITE2 LDX   #FCB     POINT TO FCB
0551 86  04                LDAA  #4       CLOSE FUNCTION
0553 A7  04                STAA  0,X      PUT IN FCB
0555 BD  D406              JSR   FMS      DO CLOSE
0558 26  01                BNE   WRITE3
055A 39                    RTS
                    * ERROR HANDLER
055B BD  CD3F       WRITE3 JSR   RPTERR
055E BD  D403              JSR   FMSCLS
0561 7E  CD03              JMP   WARMS    EXIT TO FLEX

                    * DELETE FILE
                    * This routine will delete the
                    *  "Catalog.BAK" file.
                    *  If no .BAK file is found, a return
                    *  with no error is executed.
                    ************************************
0564 0E  0820       RUBOUT LDX   #MSG15
0567 DF  CC14              STX   BFPNTR
0569 0E  CB40              LDX   #FCB     POINT TO FCB
056C BD  CD2B              JSR   GETFIL   GET FILE SPEC.
056F 25  0A                BCS   RUB1     ERRORS?
0571 86  0C                LDAA  #12      DELETE CODE
0573 A7  00                STAA  0,X
0575 BD  D406              JSR   FMS      DO DELETE FUNCTION
0578 26  01                BNE   RUB1
057A 39                    RTS
057B 0E  CB40       RUB1   LDX   #FCB
057E A6  01                LDAA  1,X      GET ERROR CODE
0580 81  04                CMPA  #4       NO FILE ERROR?
0582 26  01                BNE   RUB2     NO, SOMETHING ELSE.
0584 39                    RTS
0585 BD  CD3F       RUB2   JSR   RPTERR
0588 BD  D403              JSR   FMSCLS
058B 7E  CD03              JMP   WARMS    EXIT TO FLEX
                    * READ FILE --
                    *  This routine will read the
                    *  file "Catalog.DAT" into memory.
                    *  If no .DAT file exists, then control
                    *  is transferred to "WRITE" to write a
                    *  blank .DAT file to disk.
                    ************************************
058E 0E  0800       READ   LDX   #MSG13
0591 DF  CC14              STX   BFPNTR   BUFFER POINTER
0593 0E  CB40              LDX   #FCB
0596 BD  CD2B              JSR   GETFIL   GET FILE SPEC.
0599 25  3F                BCS   READ3    ERRORS?
059B 86  01                LDAA  #1       OPEN FOR READ CODE
059D A7  04                STAA  0,X
059F BD  D406              JSR   FMS
05A2 26  36                BNE   READ3
05A4 6F  04                CLR   0,X
05A6 86  FF                LDAA  #$FF
05A8 A7  0B 30             STAA  $B,X
                    * READ MEMORY IMAGE
05AB 0E  0C10              LDX   #TABLE
05AE DF  0102       READ1  STX   TEMP1    SAVE POINTER
05B0 0E  CB40              LDX   #FCB
05B3 BD  D406              JSR   FMS
05B6 26  0E                BNE   READ2    ERRORS?
05B8 BE  0102              LDX   TEMP1    RESTORE POINTER
05BB A7  04                STAA  0,X      SAVE IN MEM.
05BD 08                    INX
05BE 3C  7FFE              CPX   #MEMTOP  TOP OF MEM REACHED?
05C1 27  08                BEQ   READ4    DONE
05C3 8A  80                ORAA  #80      DO ANOTHER
                    * END OF FILE ERROR
05C5 0E  CB40       READ2  LDX   #FCB
05C8 A6  01                LDAA  1,X      GET ERROR CODE
05CA 81  08                CMPA  #8       END OF FILE ERROR?
05CC 26  0A                BNE   READ3    NO, SOMETHING ELSE.
                    * CLOSE FILE AND EXIT
05CE 86  04         READ4  LDAA  #4       CLOSE FUNCTION
05D0 A7  00                STAA  0,X
05D2 BD  D406              JSR   FMS
05D5 26  01                BNE   READ3
05D7 39                    RTS
                    * NO FILE ERROR
05D8 0E  CB40       READ3  LDX   #FCB
05DB A6  01                LDAA  1,X      GET ERROR CODE
05DD 81  04                CMPA  #4       NO FILE ERROR?
05DF 26  ED                BNE   READ4    NO, SOMETHING ELSE
05E1 7E  05EE              JMP   WRITE    WRITE BLANK FILE

                    * STRINGS *
                    ***********
05E4 20 20 20 20    MSG1   FCC   /         *** INVENTORY CATALOG ***/
05E8 20 20 20 20
05EC 20 20 20 20
05F0 20 20 20 2A
05F4 2A 20 49 4E
05F8 56 45 4E 54
05FC 4F 52 59 20
0600 43 41 54 41
0604 4C 4F 47 20
0608 2A 2A 2A
060B 04 00                FCB   $04,$00
060D 20 20 20 20    MSG2   FCC   /     Rev. 1.0    B. Lund    02-Sep
0611 20 20 20 52
0615 65 76 2E 20
0619 31 2E 30 20
061D 20 20 20 20
0621 52 2E 20 4C
0625 75 6E 64 20
0629 20 20 20 20
062D 30 32 2D 53
0631 65 70 2D 2E
0635 20 20 32
0638 08 0A 0A 0A    MSG3   FCB   $08,$0A,$0A,$0A,$0A
063C 0A
063D 45 6E 74 65    MSG4   FCC   /Enter item ..... (1)/
0641 72 20 69 74
0645 65 6D 20 2E
0649 2E 2E 2E 2E
064D 2E 2E 2E 2E
0651 2E 2E 2E 20
0655 28 31 29
0658 0A 00                FCB   $0A,$00
065A 44 65 6C 65    MSG5   FCC   /Delete entry ..... (2)/
065E 74 65 20 65
0662 6E 74 72 79
0666 20 2E 2E 2E
066A 2E 2E 2E 2E
066E 2E 2E 2E 2E
0672 2E 2E 2E 20
0676 28 32 29
0679 00 0A                FCB   $00,$0A
067B 53 75 62 74    MSG6   FCC   /Subtract item ..... (3)/
067F 72 61 63 74
0683 20 69 74 65
0687 6D 20 2E 2E
068B 2E 2E 2E 2E
068F 2E 2E 2E 2E
0693 2E 2E 2E 20
0697 28 33 29
069A 00 0A                FCB   $00,$0A
069C 53 75 62 72    MSG7   FCC   /Are you sure? (Y or N):/
06A0 65 20 79 6F
06A4 75 20 73 75
06A8 72 65 3F 20
06AC 28 59 20 6F
06B0 72 20 4E 29
06B4 3A
06B5 04                   FCB   $04
06B6 0A 0A 49 4E    MSG8   FCC   /ION ITEM:/
06BA 56 45 4E 54
06BE 4F 52 59 20
06C2 49 54 45 4D
06C6 3A
06C7 20 20 20 20          FCC   /         / TEN SPACES
06CB 20 20 20 20
06CF 20 20
06D1 44 45 53 43          FCC   /DESCRIPTION/
06D5 52 49 50 54
06D9 49 4F 4E
06DC 04                   FCB   $04
06DD 50 52 49 43    MSG9   FCC   /PRICE/
06E1 45
06E2 04                   FCB   $04
06E3 20 20 20 20    MSG10  FCC   /---- ---- /
06E7 20 20 20 20
06EB 20 20
06ED 04                   FCB   $04
06EE 53 75 62 74    MSG11  FCC   /Subtract how many?:/
06F2 72 61 63 74
06F6 20 68 6F 77
06FA 20 6D 61 6E
06FE 79 3F 3A
0701 04                   FCB   $04
0702 43 61 74 61    MSG13  FCC   /Catalog.DAT/
0706 6C 6F 67 2E
070A 44 41 54
070D 00                   FCB   $00
070E 43 61 74 61    MSG14  FCC   /Catalog/
0712 6C 6F 67
0715 00                   FCB   $00
```

```
081C 42 41 4B           FCC   /BAK/
081F 04                 FCB   $04
0820 63 61 74 61  MSG15 FCC   /Catalog.BAK/
0824 6C 6F 67 2E
0828 42 41 4B
082B 00                 FCB   $00
082C 53 65 61 72  MSG16 FCC   /Search for:/
0830 63 68 20 66
0834 6F 72 3A
0837 0A 08              FCB   $0A,$08
0839 53 65 61 72        FCC   /Search ..... (4)/
083D 63 68 20 2E
0841 2E 2E 2E 2E
0845 2E 2E 2E 2E
0849 2E 2E 2E 2E
084D 2E 2E 2E 2E
0851 2E 2E 2E 2E
0855 2E 2E 2E 20
0859 34 29
085B 0A 08              FCB   $0A,$08
085D 45 78 69 74        FCC   /Exit to film ..... (5)/
0861 20 74 6F 20
0865 66 6C 65 78
0869 20 2E 2E 2E
086D 2E 2E 2E 2E
0871 2E 2E 2E 2E
0875 2E 2E 2E 20
0879 35 29
087B 0A 08              FCB   $0A,$08
087D 50 72 69 6E        FCC   /Print catalog ..... (6)/
0881 74 20 63 61
0885 74 61 6C 6F
0889 67 20 2E 2E
088D 2E 2E 2E 2E
0891 2E 2E 2E 2E
0895 2E 2E 2E 20
0899 36 29
089B 0A 08 0A 0A        FCB   $0A,$08,$0A,$04,$0A
089F 0A
08A0 43 4E 54 45        FCC   /ENTER SELECTION:/
08A4 52 20 53 45
08A8 4C 45 43 54
08AC 49 4F 4E 3A
08B0 04                 FCB   $04
08B1 2A 2A 2A 20  MSG2  FCC   /*** ALL ENTRIES FULL ***/
08B5 41 4C 4C 20
08B9 45 4E 54 52
08BD 49 45 53 20
08C1 46 55 4C 4C
08C5 20 2A 2A 2A
08C9 0A 08 07 04        FCB   $0A,$08,$07,$04

08CD 51 75 61 6E  MSG3  FCC   /Quantity (99 max.):/
08D1 74 69 74 79
08D5 20 28 39 39
08D9 20 6D 61 78
08DD 2E 29 3A
08E0 04                 FCB   $04
08E1 20 20 20 20  MSG4  FCC   /        Item # :/
08E5 20 20 20 20
08E9 20 20 20 49
08ED 74 65 6D 20
08F1 23 20 3A
08F4 04                 FCB   $04
08F5 44 65 73 63  MSG5  FCC   /Description (62) :/
08F9 72 69 70 74
08FD 69 6F 6E 20
0901 20 28 36 32
0905 29 20 3A
0908 04                 FCB   $00
0909 50 72 69 63  MSG6  FCC   /Price (no $ sign) :/
090D 65 20 28 6E
0911 6F 20 24 20
0915 73 69 67 6E
0919 04                 FCB   $04
091A 2A 2A 20 4E  MSG17 FCC   /** NOT FOUND **/
091E 4F 54 20 46
0922 4F 55 4E 44
0926 20 2A 2A
0929 04                 FCB   $04
092A 45 6E 74 65  MSG18 FCC   /Enter (CR) to continue/
092E 72 20 3C 43
0932 52 3E 20 74
0936 6F 20 63 6F
093A 6E 74 69 6E
093E 75 65
0940 04                 FCB   $04

              0C1B TABLE  EQU   $0C1B
              075F COUNT  EQU   $-MSG1OP  BYTE COUNT
                          END   TABCLR

0 ERROR(S) DETECTED
```

Omegasoft Pascal Review

Tom Blank
2282 Williams St.
Palo Alto, Ca. 94306
415-858-2885

The OmegaSoft Pascal software should not be called just another compiler. It is a Pascal environment designed for efficient program development. The system is composed of the following major components: compiler, debugger, relocatable assembler, runtime library, linking loader, and good documentation. The combination of these six pieces provides the facility to efficiently create binary programs that are relocatable, reentrant, ROM-able, and that contain no unused code. Additionally, the language implementation conforms very closely to the "Pascal User Manual and Report" by Jensen and Wirth; to overcome some of the limitations of "Standard Pascal," many language extensions were added — primarily to facilitate input/output (I/O) and to aid in character string manipulation.

Upon receipt of the OmegaSoft Pascal system, the most notable observation is the inclusion of six manuals totaling 396 pages and 114 files in 1511 256-byte sectors which include the source-code of the entire runtime library. At first the size of both documentation and files seems overwhelming, but the manual (for my system), titled "Pascal Configuration Manual for Flex," begins with a simple, concise description of what files are what, what files should be on the system disk, basic program commands, and system dependencies. Additionally, the manual includes a step-by-step example which demonstrates the major features and techniques for using the system. Without reading any of the other manuals, the example ran as described with no problems.

The next discovery was the wealth of information in the OmegaSoft "Pascal Language Handbook." It is formatted like an encyclopedia: entries are arranged alphabetically. The entries range from all the supplied procedure/function names to Pascal concepts like scoping and even to the meaning of punctuation. All entries have a common format which includes a quick reference guide, syntax diagram, description of the entry, and an example demonstrating the use of the feature. Contrary to using Jensen and Wirth, the Handbook allows very quick access to the available language features, syntax, and exactly what will happen (feature semantics). The only difficulty is that any implementation change requires changing the Handbook; therefore, a supplementary handbook is supplied requiring the consultation of two books to be sure of accurate information.

Besides the conventional Pascal features — such as real numbers, integers, records, goto's, functions, procedures, etc. — which are all faithfully implemented, there are a number of important extensions to "Standard" Pascal. The easiest to understand is the addition of a string type which is a variable length array of up to 126 characters. Similar to many BASICs, routines for string matching, sub-string assignment, and string-type conversions (i.e. string into an integer) are included.

Important Pascal extensions for use on Micros are the type, byte, and the ability to define the location in memory of variables. For example, the variable ACIA could be defined as a byte-type at location $E004 (the usual location of the FLEX command port). Then all references to that variable would automatically access the command port located at that

address. Of further value to Micro users are the facilities to include assembly language routines. The conventions for both Pascal calling assembly routines and assembly calling Pascal routines are provided and throughly described.

Further expanding the normal Pascal capability are the additions for I/O. Enhancements revolve around the type called "device," which allows external system communication. Six standard devices are pre-declared for use:

INPUT - a text device for input from the command port.

OUTPUT - a text device for output to the command port.

AUXOUT - a text device for output to the system printer.

KEYBOARD - a character device for single character input from the command port (the input is not echoed).

DISK - a file device that is used to communicate with the file management system. Allowable operations are file input, output, update, and random file access.

Documentation for the creation of new device types is included so that you may use the conventional Pascal procedures/functions for I/O with your own custom device. An extensive example is included comparing the programming techniques for a memory-mapped display: first as a device type; second as a variable, fixed at the screen location.

Another significant enhancement is the capability for modular compilation. This means that separate pieces of one program may be separately compiled. At first this may seem unimportant, but it opens many time-saving possibilities. For example, a library of useful routines can be created and separately compiled. Then any program that needs to use one of the routines can use it without having to recompile it; the binary code for the routine is automatically connected to your program. Another advantage of modular compilation is that very large programs can be broken into smaller pieces that are more quickly compiled. Additionally, only the modules that change during debugging need to be recompiled — which is another time savings.

When writing a program, the first step is creating a disk text file with the Pascal program. Step two is compilation. Since debugging will usually be step three, the compiler can be instructed through a command line option that debugger code should be generated. Other compiler options are available for normal assembly code, printing, halt on errors, and for redirecting source, output, and listing files (UNIX type syntax is used). The compiler only takes one pass and is quickly done. If errors are detected, the program line in violation, an error pointer, and the description of the error are printed.

After compilation, the debugger can be called. It provides the facilities to completely regulate and monitor the execution of the program under test. The following command types are possible:

Program Execution Control - the program can be stepped line-by-line or executed until pre-set break points are hit.

Variable Display/Modification - any type of variable can be displayed or modified.

Debugger General Display - permits the display of program attributes such as stack pointers, heap pointer, and nesting level. By setting various program switches, the debugger can print each line number before it is executed, or write the name of any procedure/function before it is executed.

In comparison to a mainframe Pascal debugger (specifically the Berkeley VAX software), there is only one feature missing from the OmegaSoft debugger: the capability to examine the source program from within the debugger. This is not a major problem since a program listing in combination with the debugger provides all the capability necessary to exterminate program bugs.

After the program is debugged, an independent, executable binary file can be created. Conceptually, it is a three step process: first, the source program is recompiled to produce assembly language output (debugger output is different); second, the compiler output must be assembled to produce a relocatable binary module; third, the linking loader takes the binary program module and only the necessary routines from the runtime library module and creates a binary disk file ready for execution. Fini.

The Relocatable Assembler is worthy of further mention since it is an independent program. Before proceeding, the difference between a conventional assembler and a relocating assembler is important. In a conventional assembler, the assembly language input is converted directly into an executable binary file with all program symbols completely defined, including the starting load address of the file. Whereas, a relocating assembler does not directly produce an executable binary file or require that all symbols be defined within the body of the program. The relocatable assembler output is the input to the linking loader which resolves all symbols from the different modules and then creates the binary file.

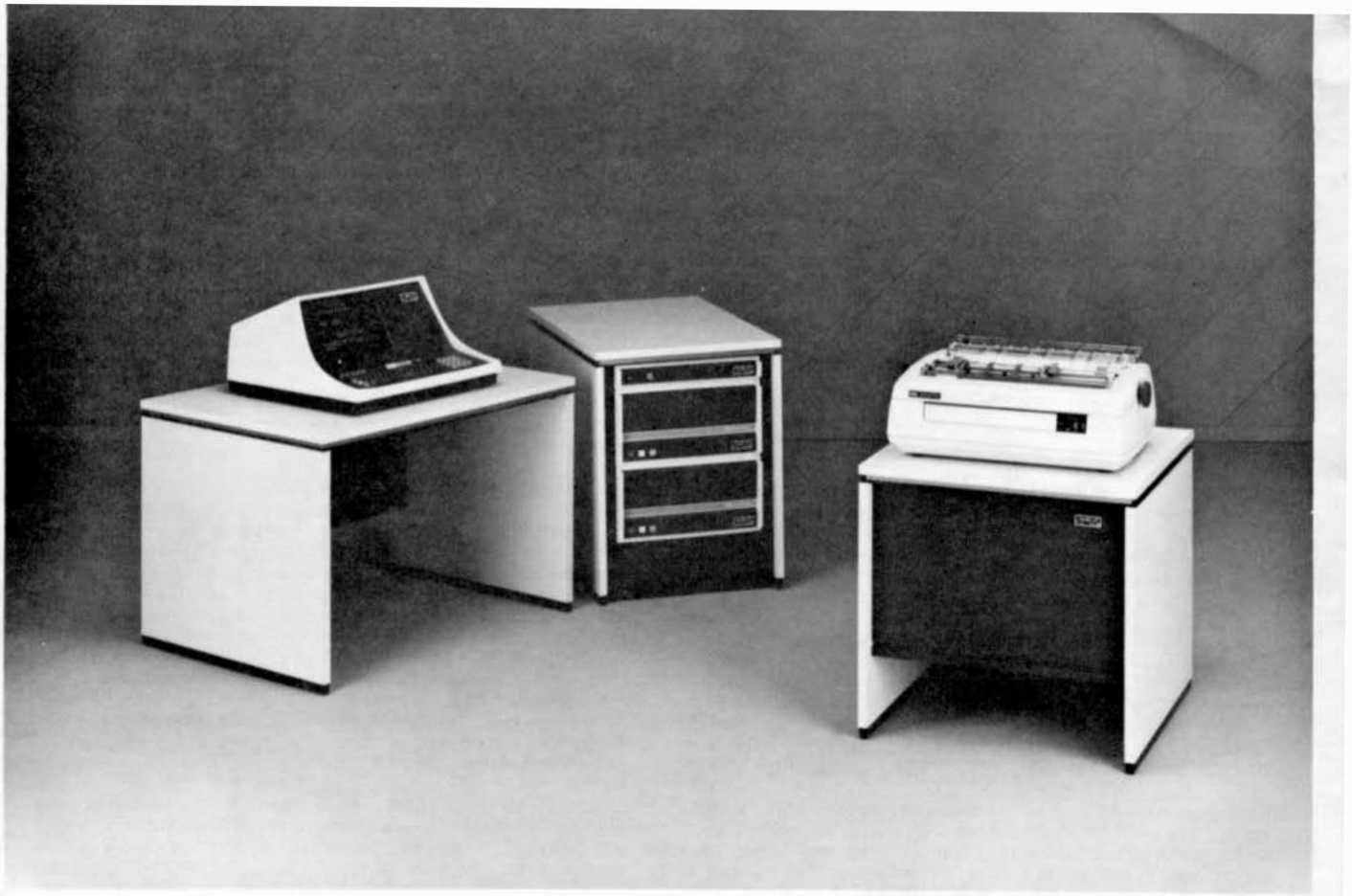
The two distinctive assembler commands are XDEF and XREF. XDEF (external definition) instructs the assembler that the following symbol should be remembered since it is an entry point and will be used by name in other programs. XREF (external reference) signifies that the symbol name is defined in another program and will be resolved later by the linking loader.

The other parts of the relocatable assembler are similar to other assemblers. For example, the system includes the following directives: FCB, FCC, FCB, etc. The only conspicuous omissions are the ORG command and macro commands which are not necessary in a relocating assembler environment.

The linking loader takes relocatable assembler output and generates an executable binary file. Commands are provided to load files, search through a library file for routines that are required (i.e. an XREF was made to the routine name), set the load address, manually define symbol names, and create the binary file. If during the binary file creation unresolved symbols (i.e. symbols that have not been defined) are encountered, error messages are printed and additional diagnostic information can be printed.

An important addition to the OmegaSoft program package was a program called the "linkage creator" (LC). Since the creation of any binary file always goes through the same steps with only changes in program names, the LC may be called to create a command file that will automatically go through the proper steps. The program is simple since it prompts for all the necessary information and always has a reasonable default answer if only a carriage return is hit. For Pascal-only users, the assembler and linking stages can be completely invisible.

The OmegaSoft Pascal compiler is priced at $425 and the combination of Relocating Assembler and Linking Loader is priced at $125 (prices given are accurate on 10/14/82). It is not unlikely that the price could discourage all but the commercial customers; however, I recommend this software package to anyone who is ready for a quality software system. It's worth the price. If performance, portability, programming speed, and the programming ease of a high-level



THE COMPLETE BUSINESS SYSTEM

+Multiuser+Highly Expandable+Cost Effective

S+ THE CONCEPT

The S+ system is a modular computer system in which all portions of the hardware and software are designed to work together in the most efficient way possible. An S+ single user system with floppy disk storage is a competitive and cost effective entry level system. Unlike most other small computers being sold as "personal", or "small business" machines, the S+ system may be expanded to maximum capabilities using this same hardware and software. You cannot end up with a DEAD END system that cannot be expanded and whose software is not compatible with larger machines. A basic S+ system may be expanded to thirty-two users, a megabyte of main memory and hundreds of megabytes of hard disk storage by simply plugging in, or connecting the desired upgrade equipment.

TOTAL DESIGN—Hardware and Software

The S+ system is an integrated hardware and software design. The two complement and enhance each other in this system. The UniFLEX® operating system used in the S+ systems is patterned after the Bell Laboratories UNIX® operating system, one of the most admired and widely used operating systems in the world. Instead of being an afterthought, the software is part of the design of the S+ system. You can be sure that with this approach that all parts of the computer operate with maximum efficiency and cost effectiveness.

THE CENTRAL PROCESSOR

The basic S+ system is configured with 256K bytes of memory and can be expanded to more than 1 million bytes. An efficient and fast hardware memory management system is used to allocate the available memory among the users on a dynamic basis. As little as 8K bytes, or the entire memory—if needed—can be used by any individual user. This makes it possible to run very large programs on the system, but it also uses no more memory than necessary for a particular job. The increase in cost effectiveness of this system over crude and outdated bank switching arrangements is dramatic.

The central processor runs in both user and supervisor states. It can detect and reject a defective user program. It is impossible for a user program to go bad and stop the entire system, as can happen quite easily in less sophisticated systems.

Task switching is accomplished by use of a multiple map RAM memory, with sixty-four individual task maps. Each task can access from 4 to 64 K-bytes of memory. Multiple tasks may be used in programs that require more than 64K bytes of memory for execution. When a task is completed the memory is automatically released for other use.

SOFTWARE

The S+ operating system, UniFLEX® is a multiuser, multitasking operating system based on the UNIX® operating system that has been used for many years on Digital Equipment Corp. PDP-11 series minicomputers. It is considered one of the most sophisticated and "user friendly" operating systems available. Variations of UNIX® are rapidly becoming standard on mini and larger microcomputers.

A large variety of languages are available for use with the system. These include FORTRAN, COBOL, BASIC, and Pascal. Word processing packages are also available to give you full text processing capability on the system.

Applications programs are available in large quantities in many fields. This includes general business, medical, dental, veterinary, library and real estate management; plus others. Since the system is multiuser it can also be connected to cash registers to produce a point-ofsale terminal system combined with the computer. The possibilities for application of this system are endless.

THE I/O SYSTEM

The S+ system is totally interrupt driven. All terminal and printer I/O devices connect to an I/O bus separate from the main bus. Up to thirty-two separate devices may be connected to the I/O bus at any one time. If I/O activity is great enough to cause an unacceptable slowdown in system operation, a separate I/O processor can be installed in the system. This plug-in option removes all I/O handling overhead from the main processor and allows operation of up to thirty-two external devices at 9,600 baud. Without an integrated total design, as in the S+ system, it would become impractical to use a UNIX®type operating system in a situation with heavy terminal I/O activity.

DISK STORAGE

A wide range of disk storage capacity is available for the S+ system, from 2.5 M-byte floppy disks to an 80 M-byte Winchester and many sizes between. All disk controllers use direct memory access (DMA) type operations to maximize data transfer and to minimize overhead on the main processor. The Winchester disks also use intelligent controllers along with DMA transfers to preserve the performance that these type devices are capable of giving. Without this distributed intelligence the system performance would be greatly degraded. The UniFLEX®operating system is designed to work at maximum efficiency with this type disk system. The data transfer rates achieved by this combination rival those of large minicomputers.

COMMUNICATIONS

A high speed local network communications system is available to interconnect S+ systems. The VIABUS® network will allow communication between systems at data rates of over 400K baud. Such a system makes it possible to share data between local systems in an efficient and low-cost manner.

AVAILABLE SOON

Tape backup—20M-Byte in less than 15 minutes on a standard ¼ inch cartridge.

Mini-Wini—5 and 10 M-Byte Winchesters—5¼ inch package. Winchester performance, for smaller systems in a small package. UniFLEX® compatible design.

Large Capacity—190 and 340 M-Byte Winchesters, plus SMD cartridge drives.

UniFLEX is a registered trademark of Technical Systems Consultants, Inc.

UNIX is a registered trademark of Bell Labs.

VIABUS is a registered trademark of Southwest Technical Products Corporation.

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 **(512) 344-0241**

compiled language are required, the Pascal system, or more aptly the Pascal environment, from OmegaSoft should be seriously considered.

Since all the features and system potential may overshadow the simplicity of using the system, an example is given that shows all the steps to generate an executable binary file. A Pascal program is also included to show many of the language features. The program is designed to change the volume name, number, and date in the system information record of a disk.

```
{ commands to create the file dskname.bin from the pascal text file and
  the assembly language routine;  my input is in upper case }

+++DA (FMSCALL )FMSCALL 0  ( assembles the prog with assem output )
+++PC (DSKNAME )DSKANEM 0  ( compiles the prog with assem output )
+++LC 'DSKNAME'            ( call the linkage creator )

Linkage Creator Version 2.1B
Copyright 1982 by OmegaSoft
Auto setup ? Y                      [ works fine for FLEX ]
System stack size : 100             [ program dependent ]
Starting load location : 100        [ you can put it anywhere ]
Library drive number : 0            [ on my system disk ]
Additional files to load : FMSCALL  [ need the assem. routine ]
Additional files to load : (CRLF)   (default none)
Additional library files : (CRLF)   (default none)
Load options : (CRLF)               (default)
Map options :  (CRLF)               (default)

+++CHAIN DSKNAME                    ( similar to FLEX exec command )

{ the file dskname.bin is now created on the working disk }

{ ************************************************************ }

 NAM FMS-CALL Assembly language routine

* FMS CALLING FUNCTION FOR USE BY A PASCAL PROGRAM
* THE CALLING DEFINITION IS:
*
*       FUNCTION FMSCAL(FCB_LOCATION:HEX):BOOLEAN
*
* WHERE THE BOOLEAN VALUE IS TRUE IF AN ERROR IS ENCOUNTERED
*
*

FMS EQU $D406

 IDEF FMSCAL SETS FMSCAL AS THE ROUTINE ENTRY POINT
FMSCAL LDX 0,U GETS FCB LOCATION FROM STACK
 LEAU 1,U MOVE FOR ONLY ONE BYTE RETURN
 CLR 0,U NORMALLY RETURN BYTE IS ZERO (FALSE)
 JSR FMS DOIT
 BNE ERROR ERROR FOUND
 RTS
ERROR INC 0,U RETURN BYTE IS ONE (TRUE)
 RTS

 END
{ ************************************************************ }

Program dskname(input,output,keyboard);

{ program to change the volume name, number and date of a disk in the
  system information record }

type
        sysrecord=  record  {see page 33 of file diagnostics;
                            the entire type is 328 bytes long to
                            overlay an FCB }
                    func_code: byte;
                    errorstat: byte;
                    actstat: byte;
                    drive: byte;
                    filename: array[1..8] of char;
                    ext: array[1..3] of char;
                    attrib: byte;
                    reserved1: byte;
                    strt_trk: byte;
                    strt_sec: byte;
                    end_trk: byte;
                    end_sec: byte;
                    filesize: integer;
                    secindir: byte;
                    reserved2: byte;
                    filemonth: byte;
                    fileday: byte;
                    fileyear: byte;
                    fcblist: hex;
                    currtrk: byte;
                    currsec: byte;
                    recordnum: integer;
                    datasecnum: byte;
                    randomindex: byte;
                    sapeerbuffer: array[1..11] of char;
                    currdirtrk: byte;
                    currdirsec: byte;
                    currdirloc: byte;
                    deldirtrk: byte;
                    deldirsec: byte;
                    deldirloc: byte;
                    scratch1: array[1..6] of char;
                    spacecoop: byte;
                    scratch2: array[1..4] of char;

                    {######## start of data in sector ##########}

                    zeros: array[1..16] of byte;
                    dskname: array[1..11] of char;
                    volnum: integer;
                    strtfree: hex;
                    endfree: hex;
                    sizfree: integer;
                    initmonth: byte;
                    initday: byte;
                    inityr: byte;
                    maxtrack: byte;
                    maxsector: byte;
                    reserved: array[40..255] of byte;
                    end;

var
        sys_fcb: sysrecord;
        newname: string[11];
        newvol:  integer;
        month:   byte at $CC0E:  {file date registers}
        day:     byte at $CC0F;
        year:    byte at $CC10:
        flag:    char;
        index:   integer;


function fmscall(fcb_location:hex):boolean;  external;

{ calls the FLEX FMS using the file control block specified. On
  return, the function is true only if the call resulted in an
  error which sets the error byte in the FCB }


begin
with sys_fcb do
   begin
   drive := $1;     { the '$' changes signifies a byte type }
   currtrk := $0;
   currsec := $3;
   func_code := $9; {read single sector}
   if fmscall(addr(sys_fcb)) then halt(errorstat+$100);
   writeln('volume name = ',dskname:8);
   writeln('volume #    = ',ord(volnum));
   writeln;
   write('Enter new volume name; 8 characters max (CR): ');
   readln(inputname);
   dskname := newname;
   for index := length(inputname)+1 to 11 do dskname[index] := ' ';
   write('Enter new volume number (CR): ');
   readln(newvol);
   volnum := newvol;
   writeln;
   write('Creation date changed to current date (y/N)? ');
   read(keyboard,flag);writeln(flag);
   if (flag='Y') or (flag='y') then
        begin
        initmonth := month;
        initday := day;
        inityr := year;
        end;
   func_code := $10; {write single sector}
   if fmscall(addr(sys_fcb)) then halt(errorstat+$100);
   writeln;
   writeln('DONE!');
  end;
end.
```

BIT BUCKET

GIMIX G-28 - SWTPC DC4

68 Micro Journal
5900 Cassandra Smith
Hixson, Tn

Don:

While developing a new SS-50 product, I discovered that the SWTPC DC-4 disk format is incompatible with other manufacturers, e.g. GIMIX G-28. Research revealed the problem to be the result of designating density on the DC-4 with SSO. The DC-4 employs the 1797 FDC which automatically verifies side. Since the side field of the sector preamble is checked against the state of SSO, it must reflect density (0=SD, 1=DD) and not side.

Correction of this problem in a DC-4 system requires both hardware and software changes. Since it can be solved in a GIMIX system with only software, this approach was selected. By maintaining a copy of the original FORMAT.CMD, the resulting system formats media compatible with both formats and automatically configures to read and write either. Perhaps the attached changes are of interest to you and your readers.

Jack D. Johnson
12219 Moorcreek Drive
Houston, Tx 77070

GIMIX G-28 SYSTEM CHANGES FOR DC-4 COMPATIBILITY

FLEX.SYS

LOCATION	OLD	NEW	SOURCE	
DEB9	848F	8B10	ADDA	#$10
DEBB	3402	84BF	ANDA	#$BF
DEBD	A621	200D	BRA	$DECC
DEBF	8EDE	8620	LDA	#$20
DEC1	3184	2033	B A	$DEF6
DEEE	B6DE	A621	LDA	1,Y
DEF0	3085	8510	BITA	#$10
DEF2	4027	27CB	BEQ	$DEBF
DEF4	08A6	8640	LDA	#$40
DEF6	218510	B4DE30	ANDA	$DE30

FORMAT.CMD (RENAME NEWDISK.CMD)

LOCATION	OLD	NEW	SOURCE	
356B	B6C10A	BD3D29	JSR	$3D29
35FA	B6C10A	BD3D29	JSR	$3D29
3682	8540	8520	BITA	#$20
36D8	8540	8520	BITA	#$20
3D26	5953	4952	FCC	'IR'
3D28	54	04	FCB	$04
3D29	454D	2707	BEQ	$3D32
3D2B	20494E	B6C105	LDA	$C105
3D2E	464F	2702	BEQ	$3D32
3D30	524D	8601	LDA	#$01
3D32	41	39	RTS	

Geoff Chapman,
70 Cliff Rd.,
Epping, 2121,
N.S.W., AUSTRALIA.

Dear Mr Williams,

The undesirability of continually inserting and removing boards from the SS-50 bus has been well documented.

In recognition of this, several ingenious methods have been described to permit switch selection of 6800 and 6809 Processors on the same bus. These have all involved wiring mods. to CPU or Motherboard, and have generally Prevented subsequent use of the DMA option.

There is a much simpler way of disabling any board plugged onto the bus. Just open the +8V rail on the card before it reaches the 5V regulator.

These links can be easily switch selected, either by a miniature SPDT toggle mounted on the top corner of each board, or by bringing all regulator input connections out to a common two position rotary switch on the front panel.

In my system, - 6809 MP-09 / 6800 MP-A Processors, 64K Ram plus 8K Rom for 6800 Monitor resters & DOS, CHAMP Disk Controller, and extended addressing Motherboard with Baud rate generator, no other board changes are necessary.

If your Motherboard is not provided with an I/O address selection switch, then this must also be added.

Ing. Herbert Strasser
Rudolf-Weisenhorn-Gasse 176
A-1238 Wien. Austria

Dear Don,

There is a small but enthusiastic 6809 Group here in Austria. Your JOURNAL is the most valuable source of information to us, and a number of System's and a lot of kits and boards have been purchased from your magazine's advertisers. Please keep up the excellent work, I look forward to every new issue of "68 MICRO".

For some reason, the January 1983 volume did never make it to my address. Please send another copy, if possible. I have the journal since October 1979 and would like to leave the collection complete.

Thank you.

Herb Strasser
OE1HSi

Paul Sanders
360 Summit Rd.
Apalachin, N Y 13732

Dear 68 MICRO JOURNAL:

I had decided not to renew as I had not found enough pertaining to the CoCo to justify continuing my subscription.

I then received the second notice and was impressed by the cute way it was written.

Not two days later I received my next issue of '68 MICRO JOURNAL, May 83 edition. And on pages 11 & 12 you saved me $72. with the Epson Converter.

I had planned to buy a ready made parallel interface for my CoCo as I have been having trouble interfacing the CoCo to 2 Centronics printers. I had tried approximately six different designs, some from books and some of my own.

I recently ran out of spare time to play around & had decided to order the interface but did not have the time to call the manufacturer when your May edition came.

I have bought most of the parts and am now redesigning the corrector portion. I hope to finish in a few days. I'll let you know how it works.

Since you have already saved me approximately $70 I will be using a portion of it to renew my subscription for another year.

entered into the computer for each specific futures contract or local market. The "Market Analyst" will perform moving average analysis including crossover analysis and make buy & sell recommendations based upon whichever crossover or moving average method you select and set up for the computer.

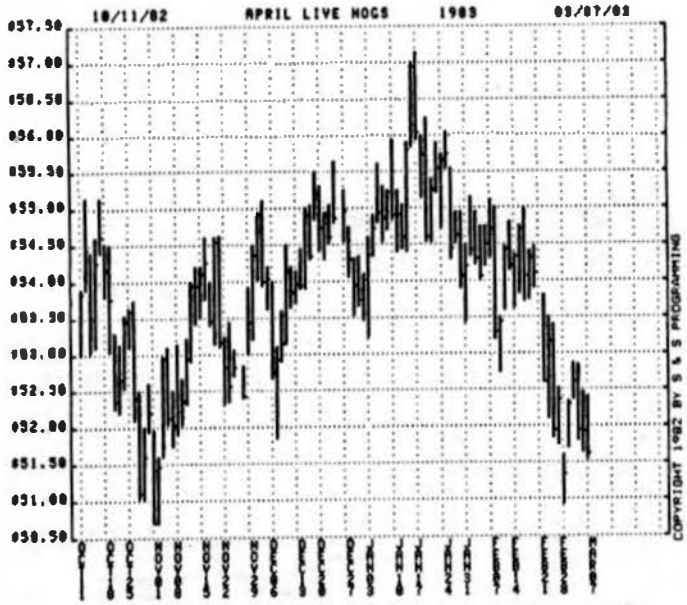
The "Market Analyst" will maintain a record of all open positions for hedge and speculative accounts. It will show a profit/loss statement for each commodity as well as a cumulative total for the year. It will monitor your account balance and margin requirements of open positions and maintain the current open equity of the account.

Updates of the program will be periodically distributed as deemed necessary by S & S Programming. As long as the original disk is returned to S & S Programming or one of it's dealers, free updates will continue to be distributed to the original purchaser of the "Market Analyst".

Parts of the package are sold individually or as a complete package. These prices are as follows:

REQUIREMENTS: FLEX (56K) or UNIFLEX (128K) system, terminal, 1 disk drive, and Epson MX-80 or MX-100 printer with Graftrax.

	FLEX	UNIFLEX
Futures Price Data Entry and Charting	$350	$450
Cash Price Data Entry and Charting	$250	$350
Basic Charting	$100	$150
Relative Strength Index Charts	$100	$150
Point & Figure Charts	$100	$150
Spread Charts	$100	$150
Moving Average Analysis	$100	$150
Futures Transactions	$100	$150
Individual Total	$1200	$1700
Complete Package	$1000	$1400



A. F. Weaver
22 Elizabeth Street
So. Burlington, Vt. 05401
(802)862-2256
May 17, 1983

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343

Dear Sos:

Enclosed is a version of the disk rename program which is very easy to use.

It is written for the 6809 and only changes the name and/or disk number. It uses the same format as the TSC RENAME utility but with the number replacing the extension.

Yours truly,

A. F. Weaver

Encl. 1

```
                    * FILE NAME "RNAMDISK"

                    * A. F. WEAVER

                    * WILL RENAME AND/OR RENUMBER A DISK
                    * USING INPUT SIMILAR TO THE FLEX
                    * RENAME UTILITY.

                    * RNAMDISK.(oldname).(00).(newname).(00)

                    * FLEX EQUATES
C940  FCBFLX  EQU   $C940
C980  SECBUF  EQU   $C980
CC02  TTYEOL  EQU   $CC02
CC11  LSTTRM  EQU   $CC11
CD03  WARMS   EQU   $CD03
CD27  NXTCH   EQU   $CD27
CD3F  RPTERR  EQU   $CD3F
CD48  INDEC   EQU   $CD48
D406  FMS     EQU   $D406

                    * OFFSETS
0010  NAMSTT  EQU   $10
001B  DSKNUM  EQU   $1B

C100                ORG   $C100
C100 20 02    RNAM  BRA   RNAM1
C102 0000     XTEMP FDB   0

                    * GET THE SIR SECTOR
C104 BD C199  RNAM1 JSR   CHKTRN
C107 1027 0099      LBEQ  SPCERR      NO INPUT!
C10B 8E C940        LDX   #FCBFLX
C10E 86 01          LDA   #1          DRIVE NUMBER
C110 A7 03          STA   3,X
C112 CC 0003        LDD   #3          TRK AND SEC
C115 ED 98 1E       STD   30,X
C118 86 09          LDA   #9          READ SING SEC
C11A A7 84          STA   ,X
C11C BD D406        JSR   FMS
C11F 1026 008C      LBNE  ERROR

                    * COMPARE DISK NAME TO INPUT NAME
C123 8E C890        LDX   #SEC.BUF+NAMSTT
C126 BD CD27  GETNAM JSR  NXTCH
C129 A1 80          CMPA  ,X+
C12B 26 08          BNE   CHKPER
C12D 8C C899        CMPX  #SECBUF+NAMSTT+8
C130 26 F4          BNE   GETNAM
C132 ED CD27        JSR   NXTCH
C135 81 2E    CHKPER CMPA #'.         PERIOD
C137 26 6F          BNE   NOFIND
C139 8C C998        CMPX  #SECBUF+NAMSTT+8

C13C 27 06          BEQ   GETNUM      IF 8 CHAR NAME
C13E A6 82          LDA   -X          DISK NAME FINISHED?
C140 81 00          CMPA  #0          NULL
C142 26 64          BNE   NOFIND

                    * GET NUMBER AND COMPARE
C144 BD CD4B  GETNUM JSR  INDEC
C147 25 5B          BCS   SPCERR      BAD NUMBER!
C149 5D             TSTB
C14A 27 58          BEQ   SPCERR      NO NUMBER!
C14C 8F C102        STX   XTEMP
C14F BE C898        LDX   SECBUF+DSKNUM
C152 BC C102        CMPX  XTEMP
C155 26 51          BNE   NOFIND

                    * IF OK PUT NEW NAME
C157 8E C890        LDX   #SECBUF+NAMSTT
C15A BD CD27  NEWNAM JSR  NXTCH
C15D 81 2E          CMPA  #'.         PERIOD
```

```
C15F 27 10              BEQ     PUTNUL
C161 A7 80              STA     X+
C163 BC C898            CMPX    #SECBUF+NAMSTT+8
C166 26 F2              BNE     NEWNAM
C168 8D CD17            JSR     NXTCH
C16B 81 2E              CMPA    #'.             PERIOD
C16D 26 05              BNE     SPCERR
C16F 20 0A              BRA     NEWNUM

                * PUT NULLS IF LESS THAN 8 CHAR
C171 4F         PUTNUL  CLRA
C172 BC C898    PUTNU1  CMPX    #SECBUF+NAMSTT+8
C175 27 34              BEQ     NEWNUM
C177 A7 80              STA     X+
C179 20 F7              BRA     PUTNU1

                * GET NEW NUMBER
C17B 8D CD48    NEWNUM  JSR     INDEC
C17E 25 24              BCS     SPCERR          SND NUMBER?
C180 5D                 TSTB
C181 27 21              BEQ     SPCERR          NO NUMBER!
C183 BF C89B            STX     SECBUF+DS)NUM
C186 8D 11              BSR     CHKTRM          INSURE LAST INPUT
C188 26 1A              BNE     SPCERR
C18A 8E C840            LDX     #FCBFLX
C18D 86 0A              LDA     #10             WRITE SING SEC
C18F A7 34              STA     X
C191 8D D406            JSR     FMS
C194 26 19              BNE     ERROR
C196 7E CD03            JMP     WARMS

                * CHECK FOR END OF LINE
C199 86 CC11    CHKTRM  LDA     LSTTRM
C19C 81 0D              CMPA    #$D
C19E 27 05              BEQ     CHKTR9
C1A0 81 CC02            CMPA    TTYEOL
C1A3 39         CHKTR9  RTS

                * ERROR ROUTINES
C1A4 86 15      SPCERR  LDA     #21             ILLEGAL SPEC
C1A6 20 02              BRA     SETUP
C1A8 86 04      NOFIND  LDA     #4              NOT FOUND
C1AA 8E C840    SETUP   LDX     #FCBFLX
C1AD A7 01              STA     1,X             ERROR STATUS
C1AF 8D CD3F    ERROR   JSR     RPTERR
C1B2 7E CD03            JMP     WARMS

                        END     RNAM
```

SMOKE SIGNAL DEALER NEWSLETTER
APRIL 1, 1983

SPECIAL DISCOUNT OFFER

In a recent article in ISC World, IBM announced a new program in which dealers could obtain an additional 1% discount when a dealer "orders exactly 440 PCs, completely filling a tractor trailer truck." Not to be out-done by IBM, SMOKE SIGNAL has determined that 150 CHIEFTAINS will exactly fill a tractor trailer truck and we offer any dealer an additional 10% discount on orders for exactly 150 CHIEFTAINS.

As an option, SMOKE SIGNAL dealers may subtract 10% from their discount on orders for exactly 2-1/2 CHIEFTAINS which completely fills the back of Ric Hammond's station wagon.

MICROWARE NEWS

Microware is planning on putting OS-9 Level II on the CRAY-1 mainframe computer. This will be an enhanced version of Level II which not only contains all the features of the present Level II, but also takes advantage of the expanded capabilities of the CRAY.

Microware also plans having available a new BASIC with FORTRAN like commands for the CRAY. A complete family of cross assemblers and compilers will be developed to facilitate transfer of 6809 based programs to the CRAY-1.

Microware intends to begin work on this project as soon as the most recent versions of Level II are completed. Ken Kaplan estimates delivery of OS-9 for the CRAY computer to begin within 30 days.

WIN BIG BUCKS IN CABINET DESIGN CONTEST

Many people have told us that it would be impossible to come up with an uglier cabinet than the CHIEFTAIN cabinet. This has angered Ric Hammond, president of SMOKE SIGNAL, since he refuses to believe anything is impossible. For the past few weeks, he has challenged the staff to come up with a less attractive design; however, no one has been able to top the current offering.

One design under serious consideration looked like a 1930 refrigerator, but this concept was vetoed by the legal department when they pointed out that a child might become trapped in the machine. Another possibility was ruled out when Ford refused to give us permission to copy the Edsel Chassis design, (used Edsel's are now too expensive to consider using the originals).

We need your help. If you have any idea how we can make an uglier cabinet, send your solution to SMOKE SIGNAL. All entries must include an 8 x 10 glossy picture or a true to scale sketch on 8 1/2 x 11 paper together with a 25 word or less description of why, in the opinion of the entrant, the proposed cabinet is, in fact, uglier than the current one. All entries become the property of SMOKE SIGNAL and the decision of the judges is final. No entries will be returned and contest is void where prohibited by local law or ordinance.

Top prize is 100,000 Argintine Pesos. (Winners from England may elect to receive an equivalent value in Polish Zlotys). All entries must be postmarked before April 30, 1983.

SMOKE SIGNAL ANNOUNCES S-100 BOARD

NO - This is not a change to the S-100 bus structure but an exciting new SS-50 product dubbed the Model S-100. SMOKE SIGNAL has retained Robert Phillips, a well known designer of SS-50 products, to design the S-100. The product contains no integrated circuits, but employs 100 DIP switches (hence the name, S-100). The board contains all standard sizes of DIP switches from packages containing only 2 switches up through 10 switches. According to designer Phillips, "this affords the user the ultimate in flexibility."

There are over a trillion switch combinations possible. It is extremely unlikely that any two users will configure their systems in exactly the same manner; thus, meeting the designer's goal of a custom system for every customer.

NEW USER FRIENDLY MANUALS

SMOKE SIGNAL announces new user manuals which accurately portray CHIEFTAIN computers in such concise and yet flowing style. They will take the beginner through detailed and easy to follow, yet enjoyable steps to using the system. These manuals will allow you to walk in, set the computer down, hand your customer the manual, and say "Have fun!" as you walk out the door.

Also from SMOKE SIGNAL are new easy to use system manuals with detailed descriptions of all components and their uses in the complete CHIEFTAIN. Each and every board, drive, power supply, and tape streamer is shown in its own exploded pictorial representation along with clearly and yet explicitly worded explanations of its function and relationship to the system as a whole.

Any change in a CHIEFTAIN computer will initiate a corresponding change in the manuals. All SMOKE SIGNAL dealers will receive immediate no charge updates to their manuals every time a change is made in the system. We feel these manuals are the epitome of accuracy!

RUMORS AND SUCH

We've heard that the staff at 68 Micro, in cleaning out an old file cabinet, discovered a file folder full of unpublished SMOKE SIGNAL new product releases and other stories, poems and submitted articles. Rumor has it that 68 Micro was astounded to discover that SMOKE SIGNAL actually manufactures MS-50 computers. Apparently some thought is being given to adding an insert to a future issue of the magazine which will contain only information on SMOKE SIGNAL products.

We've just learned that Don Williams, publisher of the "68 Micro Journal" has been retained as a consultant by the Word Processing Society of North America to define standards for future office automation and word processing. In numerous studies of speed reading conducted by Evelyn Wood, it was conclusively determined that the average person could scan a column of exactly 28.5 characters without sideways movement of the eyes. Thus, this was determined to be the optimum width for a printed column.

Drawing from this wealth of knowledge contained in the scientific literature, Mr. Williams has concluded that 28 characters should be the standard column width for all word processors. By complete coincidence, he discovered that the Radio Shack color computer has a column width of 32 columns and we expect him to recommend that this machine be adopted as the standard CPU for word processing. "Touch typists will appreciate the short keystroke which reduces finger fatigue and the 4 extra column positions provide ample room to extend beyond the normal margin in the event the intellectual user may wish to use long words", according to Mr. Williams.

Before you run off and make any corporate decisions, take a closer look at the date of this newsletter.

Editors's Note: Aw come on fellows, lets keep the record straight. Actually if you had started your subscription before March 1983, you would have noticed we ran a 'product notice' for Smoke Signal Broadcasters back in '79! Well, actually it might have been a jelly company instead, can't remember. But seems I remember you guys.

As to the 'insert' it actually is a blurb, about how the 'Indians' finally are converting over to hi-tech. But I always did wonder why you sent out a set of beads (fake turquoise) with your latest tape system.

As to the implication that the S B cabinet is ugly. Well, I want all my readers to know that I don't think so, and neither do my chickens.

On the serious side(?), actually I have run about everything (not outdated), that I have received concerning SSB. Funny thing I don't get much in the way of gripes or complaints... what's wrong???

SPEED UP CoCo Disk

772 Magnolia
St. Paul, Minnesota
55106

Don Williams Sr.
Publisher
'68 Micro Journal
POB 849
5900 Cassandra Smith Rd.
Hixson, Tennessee 37343

Dear Mr Williams,

Thanks for your follow up on my dealings with [redacted]. (Copy of your correspondence attached for reference.) [redacted] did refund the purchase price plus interest six weeks after I wrote to Visa with a copy to '68 Micro and [redacted].

I realize that mistakes can occur, but I feel that there is never an excuse to charge for a product that is not shipped! At the least this indicates sloppy business practice in a company that is selling business software. Another indication is the failure to date their correspondence. It takes more than good code and good documentation to have a well run software house.

Although the financial dealings with [redacted] have been settled, I will never do business with them again and I will never recommend them to anyone else. No mater how good their product is.

If you want to publish this letter, feel free to edit as you wish. Including removing [redacted]'s name since the problem of bad business practice seems to exist in other companies as well (this opinion is based on reading the letter column of '68 Micro and other publications). If companies realized how much harm they do to themselves by sloppy practices, perhaps there would be fewer problems.

It may interest you to know that I have heard nothing from either The Rainbow or Color Computer News on this mater (they received a copy of my letter at the same time you did). Your interest and follow up will help both users and suppliers of software.

All the best,
Bill Drew
Bill Drew

New Product Anouncement

8448K/68 CPU Board for SS-50 Systems

May 15, 1983

LSI Enterprises Ltd. announces its new MC68000 based CPU for SS-50 systems. The new board, part number 8448K/68-CPU, contains both boot-strap EPROM and a monitor EPROM, on-board timer for multi-user/multi-tasking applications, a memory management unit, vectored priority interrupts and is fully SS-50c compatible. It also features a user selectable option for a new asynchronous bus protocol that increases SS-50 bus bandwidth and is compatible with most existing SS-50c hardware.

LSI Enterprises Ltd. will license CP/M-68K from Digital Research of Pacific Grove, Ca. for the 8448K/68-CPU and soon to be released 8448K/68 system. CP/M-68K is file compatible with CP/M-80 and has commitments from many current CP/M-80 software vendors for support. This represents the first time that SS-50 users will have a native CP/M environment.

Projected prices are $795 for the 8448K/68-CPU and $850 for CP/M-68K. Shipments are expected to start July 15, 1983 for both 8448K/68-CPU and CP/M-68K. Contact LSI for updated information and prices. Write:

LSI Enterprises Ltd.
P.O. Box 1237
Woodhaven, NY 11421

or dial our 300 baud data line at 212-296-0029 (103 compatible, originate or answerback) for information. User is 8448K; password is INFO.

Dear Don,

Enclosed find a two year renewal to my 68 MICRO subscription. I originally subscribed to your magazine because I was a 6800 fan and had anticipated owning a 6800 based system (6809 or other). Well, after owning a TRS-80C for almost a year I was ready to log my 68 MICRO subscription expire in favor of an exclusively Color Computer oriented publication. However, after reading your comment in the February "Rumors" column, I have been persuaded to stick with the magazine I have come to respect as one of the best publications in the field. Please count me among those interested in more Color Computer information. I may even be interested in supporting articles or a column of some kind. I have found the TRS-80C to be very powerful when combined with the right technical expertise. I have used this computer for a variety of applications from ham radio to real time data collection. I would like to reach and hear from others who have found similar applications for their TRS-80C.

Consider this subscription renewal a vote of confidence in your commitment to Color Computer users. I'll be with you all the way.

Sincerely,

Frank J. Doran, WA3LKM
RD 1
Spring Mills, PA 16875

Meteorology Dept.
University of Utah
Salt Lake City, Utah
March 11, 1983

Mr. Robert L. May
c/o Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
P.O. Box 849
Hixson, TN

Dear Robert:

I would like to describe a hardware modification I have made to the Color Computer so it may run in the faster modes when using the disk controller. The reader may wish to refer back to my letter, describing these modes and how to enter them, in the Nov. 1982 issue of 68 MICRO on page 36.

HIGH SPEED MODIFICATION FOR DISK

When I finally bought a disk drive for the Color Computer, I found the system would "hang-up" upon trying to enter the 0.9/1.8 MHz and 1.8 MHz modes. On page 12 of the February 1983 issue of Color Computer News, Alexander Benenson suggested the removal of C73, C75, and C85 along with bypassing, if they are not ferrite beads, R73, R74, and R80. The problem with this proceedure is, since these R-C combinations act as passive low-pass filters, their removal also removes any possible filtering of high frequency transients. Examination of the Q, E, and CTS* signals with an oscilloscope showed a significant change in only the CTS* waveform when the computer was operating with the disk controller in the higher speed modes. The possible effect of changing only the R80-C85 combination was examined.

For a low-pass filter, if we define the critical frequency to be $f_c = 1/(2\pi RC)$, we find

R-C Combination	f_c
R74-C75 & R73-C73	6.05 MHz
R80-C85	1.54 MHz

Changing C85 from 220pf to 100pf and R80 from 470Ω to 270Ω will change the critical frequency from 1.54 MHz to 5.89 MHz, sufficient to permit operation in the high speed modes with a disk drive while retaining some high frequency filtering on the CTS* line.

This modification, while involving a complete disass ly of the computer, is simple and takes about one hour. The resistor R80 is located near pin 40 of U4 (6821 PIA) and the capacitor C85 is near pin 40 of the ROMPAC socket. Once the modification is completed, you have a 1.8 MHz computer since the disks will function in all three modes.

Although I have yet to try it, using the EXT command and setting $FFD9 to zero in FHL FLEX, one could conceivably operate at 1.8 MHz and still have a useable display through an external terminal!

Sincerely,
David G. McDonald
David G. McDonald

Meteorology Dept.
University of Utah
Salt Lake City, Utah
March 11, 1983

Mr. Robert L. May
c/o Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
P.O. Box 849
Hixson, TN

Dear Robert:

ANONYMOUS AUTHOR FOUND !

The March 1983 issue of Color Computer News carried an article comparing the results of a benchmark program on the Color Computer and various other mates. The Color Computer was run in the 0.9, 0.9/1.8, and 1.8 MHz modes. My letter (68 MICRO, Nov. 1982, p. 36), describing how to enter this 1.8 MHz mode, was cited but was attributed to "an anonymous author from the University of Utah". Evidently between the receipt of the letter and publishing, my name was somehow deleted.

The "anonymous author" is no longer unknown, but I am alive and well and living in Utah!

Sincerely,
David G. McDonald
David G. McDonald

Dear Don Williams,

After reading your magazine for many years and taking advantage of the wealth of knowledge of your other contributors, I am happy to offer something in return. Enclosed, are a couple of program listings which I find very useful and am sure can be used by others.

The first listing is called 'Term'. This is an interupt driven terminal driver routine with a type ahead buffer. This allows the user to continue to type commands into Flex, while Flex is busy listing programs, doing a disk seek etc... The 'init' portion of the program patches the terminal routine pointers as well as a couple other locations necessary to assure reliable operation. Provisions are made in this program to allow a polling routine in the event of multiple sources of interupts, eg. printer spooling etc... The program can be reassembled to any location, although it is recommended to keep this out of the way so as not to let this get klobered by other programs. The original reason for writing this program was to allow smooth scrolling on a VT100 type terminal, which uses ^S and ^Q to control the incoming data so control code trapping routines were incorperated. As well as ^S and ^Q which stop and start the data flow respectivly, ^Y will stop anything going on and return you to Flex's Warms entry point. ^C is an intermediate return which uses the Escape Return Register address as the reentry point. ^O and 'ESC' merely toggle ^S and ^Q to stop and start the screen display, the '^' character will allow the next character to be passed to Flex, so programs that need control characters that have been incorperated here can still be used. And finally ^T, this will print a register dump of the current processor state, this is very handy for finding out where a program is going wrong.

The next listing 'L' is a simple memory resident command which will reexecute the last command used without reloading the command off of disk. In this way, multiple files can be assembled without reloading the assembler every time or multiple directories etc...

I hope you find these programs as usefull as I have and thanks again for the publication.

Sincerely Yours,

Ken Russell
11023 40th Ave. N.E.
Seattle, Wash. 98125

TERM

```
*****************************************************************
*                                                               *
*          INTERUPT TERMINAL HANDLER                            *
*            W/TYPE AHEAD BUFFER                                *
*                                                               *
*          WRITTEN BY DAVID D. SCHOENMAKER                      *
*           AND KEN RUSSELL                                     *
*                                                               *
*****************************************************************

* EQUATES
PORT    EQU     $E004           INPUT PORT
CTLS    EQU     $13
CTLQ    EQU     $11             CONTROL Q CHARACTER
CTLO    EQU     $0F             CONTROL O CHARACTER
ESC     EQU     $1B             ESCAPE(SAME AS ^O)
CTLY    EQU     $19             ^Y IS A COMPLETE CANCEL, BACK TO OS
CTLT    EQU     $14             ^T RETURNS THE CURRENT PROCESSOR STATE
CTLC    EQU     $03             ^C IS AN INTERMEDIATE RETURN
PASCHR  EQU     '^              PASS NEXT CHAR (NO CONTROL CHR TRAP)
*
WARMS   EQU     $CD03           FLEX RETURN
ESCRET  EQU     $CC16
OUTHEX  EQU     $CD3C           FLEX ROUTINES
OUTADR  EQU     $CD45
PSTRNG  EQU     $CD1E
PCRLF   EQU     $CD24           END OF FLEX ROUTINES
FIXUP   EQU     $D3E5           FIXUP AREA OF FLEX
PRTINI  EQU     $99             INITIALIZER WORD FOR ACIA
RESET   EQU     $03             MASTER RESET ACIA
MASK    EQU     $EF             CONDITION CODE MASK . . .
IRQVEC  EQU     2               IRQ VECTOR OFFSET
IRQFLX  EQU     $CD79
INCHNE  EQU     0               VECTOR TABLE OFSETS
STAT    EQU     18
OUTCH   EQU     20
INCH    EQU     22
TINIT   EQU     14
INCHV   EQU     $CD0A
INCH2V  EQU     $CD0D
OUTCHV  EQU     $CD10
OUTC2V  EQU     $CD13
*
* INITIALIZING ROUTINE
* INITIALIZES VECTORS, SETS INTERUPTS, FIXES FLEX UP
* POINT TO NEW I/O
*
        ORG     $E4DD           RUN IN HIDDEN MEMORY
INIT:   ORCC    #$10            TURN OFF INTERUPTS
        LDX     #FIXUP          I/O VECTOR TABLE
        LDD     #GETCNE         NO ECHO INPUT ROUTINE
        STD     INCHNE,X
        LDD     #IRQ
        STD     IRQVEC,X
        STD     IRQFLX
        LDD     #INIT
        STD     TINIT,X
        LDD     #INTST
        STD     STAT,X
        LDD     #PUTC
        STD     OUTCH,X
        STD     OUTCHV          ALS  FIX UP USER JUMPS
        STD     OUTC2V
        LDD     #GETC
        STD     INCH,X
        STD     INCHV           MORE USER JUMPS...
        STD     INCH2V
*
* INITIALIZE THE PORT . . .
* RESE  IT AND THEN ENABLE INTERUPTS
*
        LDA     #RESET          MASTER RESE
        STA     PORT            . . .
        LDA     #PRTINI         LOAD PORT INITIALIZER . . .
        STA     PORT            AND SET IT.
*
        ANDCC   #MASK           MAKE SURE INTERUPTS ARE ENABLE
        JMP     WARMS           RETURN TO THE MONSTER . . .
*
IRQ     LDX     #PORT           POINT TO ACIA PORT
        LDB     ,X              GET STATUS
        ASRB                    ROTATE OUT THE INSTATUS BIT
        BCS     RINT            DO IT IF SET
        RTI                     THIS LOCATION CAN POINT TO THE NEXT
*                               INTERUPT POLL ROUTINE LOCATION
*
RINT    LDA     1,X             GET THE CHARACTER
        ANDA    #$7F            CLEAR MSB SO WE DONT GET MESSED-UP
        CLRB
        CMPB    PASSF           TEST IF PASS
        BNE     SAVEIT          YEP . . .
        CMPA    #PASCHR         CHECK IF PASS CONTROL CHR
        BEQ     PASS
        CMPA    #$1F            IS IT A CONTROL CHAR
        BLE     CNTRL           YEP
*
SAVEIT  LDX     INPTR           NOW POINT AT BUFFER QUEUE
        DEC     INCNT           BUMP THE CHAR COUNTER
*                               NOTE WE COUNT NEGATIVE DIRECTION
        STA     ,X+             SAVE THE CHARACTER
        CLR     PASSF
        CMPX    #ENDING         END OF QUEUE
        BNE     RET             NOPE . . .
        LDX     #IND            RES T BUFFER POINTER
RET     STX     INPTR
        RTI                     DONE . . . WE DID OUR PART !!
*
PASS    COM     PASSF           NEXT CHAR WILL PASS, NO CONTROL TRAP
        RTI
*
CNTRL   CMPA    #CTLS           CHECK FOR CNTL S
        BEQ     S

        CMPA    #CTLO           CHECK FOR CNTL O
        BEQ     O
        CMPA    #C LO           CHECK FOR CNTL O
        BEQ     O
        CMPA    #ESC            ESCAPE IS THE SAME AS ^O
        BEQ     O
        CMPA    #CTLC           ^C
        BEQ     C
        CMPA    #CTLY           ^Y
        BEQ     Y               COMPLETE BOMB....
        CMPA    #CTLT
        BEQ     T
        BRA     SAVEIT          OTHERWISE TREAT AS ANY OTHER
*
S       LDA     #$FF            FF MEANS PAUSE
        STA     PSFLAG          SET THE FLAG
        RTI
*
O       CLR     PSFLAG          CTL O ALWAYS CLEARS FLAG
        RTI
*
O       COM     PSFLAG          CTL O DOES BOTH
        RTI
*
* FIX UP STACK AND EXIT
Y       LDX     #WARMS          GET WARM START ADDR
        STX     $0A,S           REPLACE IT ON STACK
        CLR     PSFLAG
        RTI
*
C       LDX     ESCRET          GET INTERMEDIATE RETURN
        STX     $0A,S           REPLACE IT ON STACK
        CLR     PSFLAG
        RTI
*
* CONTROL T PRINTS CURRENT PROCE SOR STATE WHEN THE
* INTERUPT OCCURED . . .
*
T       ANDCC   #MASK           LET INTERRUPTS OCCUR ...
        LDX     #REGMSG         SET UP LABEL FOR REGI TER DUMP
        JSR     PSTRNG
        JSR     PCRLF           GET NEW LINE FOR REGISTERS
        LDB     ,S              CONDITION CODE
        LDX     #8
ARND    ASLB
        LDA     #$30
        ADCA    #0
        BSR     PUTC
        LEAX    -1,X
        BNE     ARND
        BSR     $PC
        LDB     #3              PRINT 3 BYTEWIDE REGS
        TFR     S,X             MOVE STACK TO X REG
```

```
ARND1   LEAX    1,X
        JSR     OUTHEX          A, B, DP
        BSR     SPC
        DECB
        BNE     ARND1
        LDB     #4              COUNT FOR 4-16 BIT REGS
ARND2   LEAX    1,X             OUTADR INC X BY 1
        JSR     OUTADR
        BSR     SPC
        DECB
        BNE     ARND2
        JSR     PCRLF
        RTI                     FINALLY DONE . . .
*
SPC     LDA     #$20            LOAD A SPACE &
        BRA     PUTC            PRINT IT
*
* END OF I TERUPT HANDLING CODE
* NOW WE HAVE A GETC ROUTINE TO GET A CHARACTER FROM THE
* INPUT QUEUE
*
* GETCNE  WILL NOT ECHO INPUT CHARACTER
*
GETCNE  PSHS    X               STICK IT AWAY . . .
TEST    TST     INCNT           CHARS IN THE BUFFER ???
        BEQ     TEST            NO . . . WAIT
*
        LDX     OUTPTR          GET POINTER INTO BUFFER
        LDA     ,X+             CHAR INTO A . . .
        INC     INCNT           BUMP COUNT BACK TOWARD ZERO
        CMPX    #ENDING         END OF THE QUEUE
        BNE     RETRN
        LDX     #ING            RESET THE POINTER
RETRN   STX     OUTPTR          AND SAVE IT . . .
        PULS    X,PC            RETURN
*
* SIMPLE TEST OF INPUT - IMMEDIATE RETURN
*
INTST   TST     INCNT
        RTS
*
* GETC WILL ECHO INPUT
*
GETC    BSR     GETCNE          INPUT CHAR W/ECHO
*
* AND A SIMPLE PUTC THAT CHECKS PSFLAG TO WAIT . . .
* NON-QUEUED FOR NOW . . . ONE PROBLEM AT A TIME.
*
*
PUTC1   PSHS    B,X             SAVE EM . . .
        LDX     #PORT
WAIT    TST     PSFLAG          ARE WE PAUSED ? ? ?
        BMI     WAIT
        LDB     0,X             NO WAIT . . . GET STATUS
        ASRB
        ASRB
        BCC     WAIT            AFTER ALL THAT THE PORT NOT RDY
        STA     1,X
        PULS    B,X,PC          RETURN
*
* DATA AREA
* THE QUEUE IS A SIMPLE CIRCULAR QUEUE . . . TWO POINTERS
* AT THE HEAD POINT TO NEXT CHAR TO BE PUT IN AND NEXT CHAR
* TO BE TAKEN OFF.  A COUNT ACTS AS A CHARACTER AVAILABLE
* FLAG.  THE METHOD USED WILL NOT TAKE INTO ACCOUNT  WHEN
* CHARS ARE PUT IN FASTER THEN TAKEN OFF - - - THE INPTR
* WRAPS AROUND AND PASSES THE OUPTR
*
RTS     RTI

REGMSG  FCC     /  CC    A  B  DP  X    Y    U    PC/
        FCB     $4
PASSF   FCB     0               CONTROL CHAR PASS TO FLEX IF <>0
PSFLAG  FCB     0
INCNT   FCB     0
INPTR   FDB     ING
OUPTR   FDB     ING
ING     RMB     64
ENDING  FCB     $4
        END     INIT
```

```
TABLE   FCC     'L'             set up command table
        FCB     0
        FDB     #MAIN
        FCB     0,0,0

MAIN    JMP     [TRNSFR]        jump to test transfer address
        END
```

```
*****************************************************
*                                                   *
*          'L'    (MEMORY RESIDENT COMMAND)         *
*                                                   *
*       THIS ROUTINE WILL RE-EXECUTE THE LAST       *
*       COMMAND CALLED. SINCE THE COMMAND STILL     *
*       RESIDES IN MEMORY, THE NEED TO RELOAD THE   *
*       COMMAND IS ILLIMANATED, SAVING TIME AND     *
*       DISK WEAR. THE CALLING CONVENTION FOR THIS  *
*       ROUTINE IS THE SAME AS IS NECESSARY FOR     *
*       THE COMMAND THIS ROUTINE IS CALLING EXCEPT, *
*       AN 'L' (for 'last') IS USED IN PLACE OF THE *
*       ACTUAL COMMAND NAME. ALL PERIMETERS REQUIRED*
*       BY THE SPECIFIC COMMAND REMAIN THE SAME...  *
*                                                   *
*       WRITTEN BY KEN RUSSELL FOR FLEX9            *
*                                                   *
*****************************************************

TRNSFR  EQU     $CC1E

        ORG     $CC12
        FDB     #TABLE          set up memory resident command
*                               pointer (out of the way)
        ORG     $E400
```


FILE FIND (OS9)

Steve Childress
c/o Consejo Computer Products
31220 La Baya Drive Suite 110
Westlake Vlg, CA 91362

May 13, 1983

Dear Don & OS9-Types,

With the advent of OS9 and Winchesters, on occasion I've found that I cannot recall just where some doggone file was placed on the disk. Worse yet, my feeble mind sometimes forgets what name I used for the little bugger. So here's a little diddle I quickly wrote which helps out in these situations. The enclosed BASIC09 listing may be useful to '68 readers who are not assembly language experts but have had a close encounter with the lost file scenario.

Incidently, I'd appreciate it if the OS9 users in Southern California please would please drop me a line at the above address. Between the 5850 and the Apple 'WILL/6809/OS9' users, a collaboration may likely be practical. So check in! Thanks.

Sincerely,

Steve

```
PROCEDURE findfile
0000    (*
0083    (* "FINDFILE"    an OS9/BASIC09 utility to find a given file's home
0046    (* directory(s) if its approximate name is known maybe.
007D    (* Shell command:BASIC09 #10k FINDFILE
00A3    (* By Steve Childress, 5/83.
00BF    (* Public Domain Rights to this progam are granted.
00F2    (* No reportus buqus amongus if Ye pleasus.
011D    (* (speed? 721 files searched in 10 minutes; so don't lose files!)
015F    (* Should work on any OS9 version since OPEN /dev# is not used.
019E    DIM filecount,foundcount:INTEGER
01A9    DIM d(30):BYTE
```

```
01B5        DIM name,dev$STRING
01C0
01C1        PRINT  \ PRINT
01C5        INPUT "Enter approximate filename to search for: ",name
01F7        IF name="" THEN  \ END  \ ENDIF
0207        INPUT "Enter device name on which to look, eg. /H0 or /D0/CMD0 : "
              ,dev
0249        IF dev="" THEN  \ END  \ ENDIF
0259        filecount=0 \foundcount=0
0267        RUN lookfor(dev,name,filecount,foundcount)
0288        PRINT
0282        IF filecount<>0 THEN
028E          PRINT filecount;" files & directories are on device "; dev
02BD        ELSE
02C1          PRINT " No files suspect that '"; dev; "' contains typo."
02F5        ENDIF
02F7        PRINT foundcount;" filename(s) found similiar to '"; name; "'"
0327        END
0329
PROCEDURE lookfor
0000        PARAM devname,name$STRING,files,matches$INTEGER
0015        DIM d,dir$BYTE
0020        DIM i$INTEGER
0027        DIM dirx(32)$BYTE
0033        DIM tempname,*tempdir$STRING[128]
0043
0044        ON ERROR GOTO 4000 \(* Just exit if no such directory
006B        OPEN #d,devname$READ*DIR
0077        ON ERROR
007A        PRINT "Searching directory: '"; devname$; "'"
009C        REPEAT
009E          GET #d,dirx
00AB          IF dirx(1)<>0 THEN
00B4            i=1 \tempname=""
00C4            REPEAT
00C6              tempname=tempname+CHR$((LAND($7F,dirx(i)))
00DA              i=i+1
00E5            UNTIL dirx(i-1)>$7F
00F7
00F8            IF LEFT$(tempname,1)<>"." THEN
0108              files=files+1
0113              *tempdir=devname+"/"+tempname
0123              IF SUBSTR(name,tempname)<>0 THEN
0133                matches=matches+1
013E                PRINT
0140                PRINT "Possible match: "; tempdir; "       <<<<<<"
0168                PRINT
016A              ENDIF
016C              gLW lookfor(tempdir,name,files,matches) \(* RECURSE MYSELF
0197 1000         ON ERROR
019D            ENDIF
019F          ENDIF
01A1
01A2        UNTIL EOF(#d)
01AB        CLOSE #d
01B1 4000 END
01B6        (* ---------------------  ###  ----------------------
01F5
```

Ben Williams Mr.
'68' Micro Journal
POB 849
5900 Cassandra Smith Road
Hixson, Tenn. 37343

Bill

About a year and a half ago I purchased from Ackerman Digital Systems their EBUG 6809 CPU board and their 8-188 Prom Blaster board. From Netronics I purchased their James-IB 64K byte dynamic ram board. I have the ram board set up for 48K bytes of memory.

All three boards have performed admirably. The PROM BLASTER board can prgrm much prgrms most EPROMS on the market. The JAMS-IB ram board has never missed a bit, and works very well with the ADS EBUG CPU board.

The PROM BLASTER board as it comes from Ackerman's has a 25 volt switching regulator circuit. In order to program the newer EPROMs, one must add a small switch and a 120R ohm resistor to the board. The 120R ohm resistor, through the switch, is paralleled across the board's 47Ω resistor. When the switch is then thrown, the switching regulator will generate 25 volts for programming the newer EPROMs. In this manner, one can now program older style EPROMs like the 2708's and 2716's and by using the switch 2716's w. 2732's, and 2764's etc. can be programmed.

The nicest advantage that Ackerman's PromBlaster provides, is being able to take programs in 2708's or 2716's and them reprogramming 2732's and 2764's with the programs. Since the board is configured for different EPROMs via software, one does not have to purchase expensive EPROM modules or worry about losing the modules. The PromBlaster board already comes with a 28 pin socket for EPROMs, so when the 27128's become more available, one can readily program them too.

I like the boards, they work very well together.

The Netronics JAMS-IB ram board is a very nice ram board. It uses the INTEL 8202 dynamic ram controller chip. This chip makes designing and setting up dynamic ram boards a snap. Troubleshooting is simple, as there is only one chip, instead of 13 or twenty, to worry about.

Ackerman Digital Systems, Inc.
118 North York Road, Suite 208
Elmhurst, Illinois 60126

Netronics, R&D Ltd.
333 Litchfield Road
New Milford, Ct. 06776

Sincerely,

Earl W. Bollinger

Earl W. Bollinger
10818 North 91st. Ave.
Peoria, Arizona 85345

CLASSIFIED ADVERTISING

Two CAL-COMP 8" Disk Drives, $390.00.
Call Tom (615)842-4601.

***

For Sale: Complete Gimix 6800 System. Includes 64K, 2 disks, video monitor, keyboard, FLEX, XBASIC, XFORTH, SCREDITOR, etc.
Call (312) 736-8618 after 6 PM.

***

SSB SS-50c 2MHz 32K RAM Factory Warranty $225, Digital Research 2MHz 16K RAM $125, TM100-2 DS-DD 5" Disk $175, All work fine; SWTPC System: Chasis, MB, Pwr, 6800 cpu, Ser-IO (Ser-IO doesn't work) $150 or b.o.; Mark I modem plus schematics (receive doesn't work) b.o.; Infoton terminal old-but works fine $300+ shipping.
Robert Field (408)423-4361 7:30-9 am, 10-11:30 PM, 115 McPherson st., Santa Cruz, Ca 95060.

***

Smoke Signal 4-Meg Hard-Disk system, DS-DD 5" Disk, 64K etc., OS9-LI newest release, Introl C-1.4, Basic09, Asm, Edit, Debug, much more; System just upgraded-SSB warranty; Worth over $9000 Sell for $6900.
Robert Field, (408) 423-4361, 7:30-9 AM, 10-11:30 PM, 115 McPherson st., Santa Cruz Ca 95060.

***

For Sale: Data Systems 68 6809 CPU card-$95, F&D MDC-1 Disk controller-$85, Many other items-Write for details.
Roger Wileman, 160 Helen Way, Escondido, Ca 92025 (619)746-2934.

***

F&D MDC1 gold conn. $125 else need patches for FLEX 2.7.3; 6800D2, 6800AB, 32K, $75 or bo.
Call collect (603) 679-8410.

Software

For Ordering Call TOLL FREE 1-800-338-6800

FLEX™ OS-9™ Color Computer

Computer Systems Consultants:

AVAILABLE NOW!!!

The Powerful Super Sleuth Disassembler for the Data-Comp, Hogg, and Spectral Color Comp. FLEX Systems;

Color FLEX Object Code only $50.00
Color FLEX w/ Source Code $99.00
Normal FLEX w/ Source Code $99.00
UNIFLEX w/ Source Code $100.00
OS-9 w/ Source Code $101.00

ALL Comp Sys Cons Programs run on the Color FLEX Systems

Great Plains Computer Co.

AVAILABLE NOW!!!

STYLOGRAPH 2.0 for the Data-Comp and Hogg Color FLEX Systems. Full screen display and editing (i.e., what you see is what you get); supports proportional printers. Operates with the 51 x 24 screens.

SPECIAL COLOR FLEX STYLO $195.00
FLEX STYLO $295.00
UNIFLEX STYLO $395.00

Fast SPELLING CHECKER; allows directly changing the Text File, adding words to the dictionary, etc. 75,000 words in less than 400 sectors (easily fits 5¼" disks with other normal commands on it also).

FLEX, Color FLEX, OS-9 $125.00
UNIFLEX $175.00

MAIL MERGE — greatly extends the power and flexibility of STYLOGRAPH. Allows Multiple Text files to be printed out as one large document. Provides for merging information into the Text File during printing (such as different names and addresses), etc.

FLEX, Color FLEX, OS-9 $145.00
UNIFLEX $195.00

INFOMAG Data Base Management System — An XBASIC-Based, Menu Driven, DBMS with "Built-in" Audit Tracking, Extremely Powerful Report & Format Capabilities, etc. This "Time Proven" DBMS will become the "Work Horse" of your Software "Stable."

FLEX and Color FLEX $295.00
UNIFLEX $395.00

Also in Stock; **Accts Rec., Accts Payable & Gen Ledger** — A full blown Accounting Package that can be used together, or as separate packages; provides the IRS required Audit Tracking. (XBASIC, based on the "Osborne Business Programs.")

FLEX and Color FLEx $295.00/PROG.
UNIFLEX $395.00/PROG.

Also in stock; **PRMAC**; a Relocatable, Recursive Macro-Assembler and Linking Loader for the 6809. An enhancement of Ed Smith Software Works original system. Use either Motorola Format or Special Ed Smith Format. Supports Recursive Macros, Conditional Assembly, etc. All I/O thru normal FLEX System Calls. Can provide a full Cross Reference Listing. Includes a Small Line-oriented Editor as part of the Assembler. Greatly improved operating manual.

FLEX & Color FLEX 6809 PRMAC w/Link & Editor $150.00
SOURCE $ 50.00

Call for information on any requirement—
We have over 200 FLEX, Color FLEX, UNIFLEX, OS-9 and Color Computer Programs available; Games, Programming Languages, Business Software, etc., etc., etc. ??? More Info; just watch these ads!!!, write or call South East Media.

Depend On South East Media

Make Your 'BACK-UP' On South East Media

FREE DISKETTE With Each $50 Purchase

For Ordering Call TOLL FREE 1-800-338-6800

SOUTH EAST MEDIA
5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

O-F

Finally the barrier has been removed from OS9™ to FLEX™ formatted disk!! Now you can **READ** and **WRITE** to a FLEX™ diskette, 5 or 8 inch, with **O-F**.

O-F is a new and unique program, written in BASIC09™ that performs the following functions, and comes complete with source.

1. **REFORMAT:** This module formats a disk that can be read by both OS9™ and FLEX™. Eight or five inch selectable.
2. **FLEX.BAS:** This program does the actual read or write function to the special O-F disk. Also it has the disk format and **DIR** (OS9) commands. All selectable from a user-friendly menu. All selections are interactive and complete including all necessary prompts to the operator.
3. **DIR:** This module (menu selected) allows the disk directory to be printed to the screen, while in BASIC09.

FLEX users can read, write and use the special disk as any other FLEX disk, provided the FLEX directory is not allowed to continue beyond track zero (too many files).

$79.95

WINCHESTER BACKUP UTILITIES

The following utilities allow the backup of any size disk system to any size diskette.

By simply inserting diskettes as requested by COPY. MULT, a large disk system (Winchester, etc) may be downloaded to your present floppy disk system, any size. No need to fiddle with directory deletions or any of the other tedious operations that must be done using a normal copy routine.

COPYMULT-CMD understands normal "copy" syntax and always keeps up with files already copied by maintaining directories for both host and receiving disk system, thus eliminating hours of tedious keyboard entries and other time consuming cleanup chores.

BACKUP-CMD is a special program that downloads "random" type files, **any size.**

RESTORE-CMD a special program to restructure copied "random" files for copying, or recopying back to the host system.

FREELINK-CMD a "bonus" utility that "relinks" the free chain of a floppy or hard disk thereby eliminating fragmentation.

**Completely documented source files included. **ALL 4 Programs 99.50 on 8" diskette

CHESS 6809

Requires FLEX™ and **Now Runs On Any Type Terminal**

Features:

- Two display boards.
- Change skill level.
- Swap sides.
- Point scoring system.
- Four levels of play.
- Stop 'Mate in 1-2-3-4' moves.
- Make move and swap sides.
- Play white or black.

$79.95 Specify 5" or 8" disk

This is one of the strongest CHESS programs running on any microcomputer, estimated USCF Rating 1600+.

DIET-TRAC Forecaster

DIET-TRAC Forecaster is a program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G %) or grams of Carbohydrate. Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individuals are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the invididual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calender with each day's weight for a 30-day period is printed.

FLEX VERSION — $59.95 UniFLEX VERSION — $89.95

A COLOR COMPUTER TERMINAL DRIVER

TERM is a new and long needed terminal driver for those color computer users who have developed stinging, red and watering eyeballs! TERM allows you to switch from the CoCo keyboard and TV monitor to a real CRT video terminal. Think what that means! No more 32 or even 51 characters per line, that are so blurred that you must guess as to what some characters actually are. No more squinting or cussing. Sounds great? Well TERM gets you away from all that. TERM is called from the CoCo as any other program. You then type the 'RETURN' key on the CRT video terminal keyboard and TERM configures the CRT video terminal to the proper baud rate and you are free of squinting and guessing! To return to the CoCo keyboard (God forbid), you simply type in the command EXTERM from the CRT video terminal keyboard, and you are back where you started from, squints and all.

TERM functions from Data-Comp FLEX directly and requries only a standard CRT video terminal (any el cheapo will do) or a deluxe terminal, either works just fine. The terminal is connected to the serial port of the CoCo by a standard cable and connector. TERM does NOT function in Radio Shack mode (must be FLEX).

If you want the luxury of 80 characters by 24 lines, or more, depending on the CRT video terminal used, then **TERM** is a must!

ONLY $19.95 FREE with purchase of F-Mate

SOUTH EAST MEDIA 5900 Cassandra Smith Rd., Hixson, TN 37343 (615) 842-4601

SPELLB "Computer Dictionary"

No more "Let your fingers do the walking through the Dictionary" while you are inputting Text with your favorite Editor or Word Processor. **SPELLB**, written by Dan Farnsworth of PALM BEACH SOFTWARE, is more than "another Spelling Checker"; it allows you to "look up a word" from within your Editor or Word Processor so that you KNOW it is right WHEN YOU TYPE IT IN (if your Editor supports a method of passing a Command to FLEX without exiting the Program) with the **SPH.CMD** Utility (which operates in the FLEX Utility Space). Yes, it ALSO allows you to check and update the Text after you are finished; along with allowing you to ADD WORDS to the Dictionary, "Flag" questionable words in the Text for evaluation later, "View" a word in context before changing or ignoring, etc. **SPELLB** first checks a "Common Word Dictionary", then the normal Dictionary, then "Personal Word List", and finally, any "Special Word List" you may have specified. **SPELLB** also allows the use of Small Disk Storage systems.

FLEX and Color FLEX $129.95

"JUST" Text Formatter

JUST, a "Text Formatter" developed by Ron Anderson, provides numerous features which make it a valuable addition to any FLEX Users Software Library. JUST is designed for formatting Text Output for Dot Matrix Printers and provides many unique features:

Output the "Formatted" Text to the Display for format analysis and change.

Output the "Formatted" Text to a Text File for use with the supplied **FPRINT.CMD** for producing multiple copies of the Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (this Utility useful at other times also, and worth the price of the program by itself).

"User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); provides for up to ten (10) imbedded "Printer Control Commands", such as Italics on and off, Boldface on and off, etc.

Automatic compensation for a "Double Width" printed line.

Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc.

Use with ANY Editor.

Supplied with "Structured Source" (Windrush PL/9); easy to see the flow of the program.

FLEX and Color FLEX $49.95

PROGRAMMERS (both NEW and OLD-TIMERS) NOTE —

The Power and Versatility of the 6809 CPU has stretched the capabilities of the "Old Standard" Programming Practices and Procedures, and opened the door for the "Innovators" of the Industry to develop **NEW** and **BETTER** Software Development Tools. **WINDRUSH MICRO SYSTEMS** has stepped forward with three (3) new Programs that will find a home on **EVERY** Programmers work bench.

MACE — XMACE By Graham Trott

MACE is a combined Editor/Assembler designed to allow the Programmer to Enter, Edit, and Assemble Programs with a minimum of effort. **MACE** is designed primarily for the EASY development of small to medium sized Assembly Language programs, but larger programs can be developed using the "Spool" capabilities. The Editor (a simplified Line Editor streamlined for this package) "codes" each Op-Code, providing minimal memory requirements. **MACE** is very "forgiving", which, when combined with the "Interactive" operation, makes this an EXCELLENT package for the Beginning Programmer!

FLEX and Color FLEX — $98.00

PL/9 — By Graham Trott

PL/9 is an Editor/Compiler/Debugger all combined into ONE PACKAGE, which was devised specifically to allow the Assembly Language Programmer the "Best of All Worlds". It allows the Programmer to use "Structured Programming Techniques" while working at the Assembly Language level in a totally **INTERACTIVE** Program Development Cycle (just like working with BASIC; enter some code, try it, edit the code, try it again, etc.). The Single Pass Compiler supports up to 127 Character Symbols; Variable Types; Pointers; Control structures built around the 'Procedure' System. IF. .THEN. .ELSE, BEGIN. .END, WHILE, REPEAT. .UNTIL structures, etc., along with Stack, A-, B-, and D-Register manipulation etc. The Editor/Assembler are similar to the **MACE** Program. The Trace/Debugger is oriented towards the **PL/9** Source Program and provides Single Stepping, Breakpointing, running a specified Line Number Range, etc. All in all, this provides an excellent Software Development Tool for utilizing the power of the 6809.

FLEX and Color FLEX — $198.00

C —

Finally, for the "Big Time Operator", or for the beginner who can "see the future" and wants to become a part of it, here is the EXCELLENT **WINDRUSH MICRO SYSTEMS** "C Compiler". This is one of **THE** C Compilers for the FLEX Operating System. It can be used with normal Assemblers for most Programming, or with the TSC Relocating Assembler/Linking Loader for those "full blown" System Packages.

FLEX and Color FLEX — $295.00

SOUTH EAST MEDIA

5900 Cassandra Smith Rd., Hixson, TN 37343 **(615) 842-4601**

FOR ORDERING CALL TOLL FREE 1-800-338-6800

SOUTH EAST MEDIA Software

5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601

FLEX™ OS-9™ Color Computer

Lucidata

AVAILABLE NOW!!!

PASCAL Language and P-Code Compiler, Version 3. Designed especially for Micro Computer Systems; Run-time System checks available resources for each task, allowing operation on even minimal computer systems. Based on the proposed ISO Standard. Allows linkage to Assembler Code for maximum flexibility.

FLEX and Color FLEX 5" $190.00
FLEX 8" $205.00

PASCAL UTILITIES — Require LUCIDATA Pascal Ver. 3. **XREF** — will produce a Cross Reference Listing of any text; aimed specificity towards Pascal Source. **INCLUDE** — allows the inclusion of other Files in a Source Text; has unlimited nesting capabilities. Also allows Binary File Inclusions. **PROFILER** — produces an Indented, Numbered, "Structogram" of a Pascal Source Text File. Allows viewing the overall structure of large programs, and provides clues as to the integrity of the program. Supplied as Source Code; requires compilation.

FLEX and Color FLEX — Each program $25.00

COPYCAT — Allows reading TSC Mini-FLEX, SSB DOS68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform Miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Includes Utilities to List Directories, Copy Files, and convert Text Files when required. Also includes a Utility for investigating Physical Compatibility problems. Programs supplied in Modular Source Code to make it easier to solve unusual problems.

FLEX and Color FLEX 5" $50.00
FLEX 8" $65.00

Westchester Applied Business Systems

AVAILABLE NOW!!!

XDMS Data Management System. Possibly one of the most powerful DMS's available. This machine language program is small enought to operate on a single sided 5" disk, yet provides the speed of M.L. and power limited only by the user's imagination. Supports Sequential, Hierarchical, and Random Access File Structures, and has Virtual Memory capabilities for those Giant Data Bases. Easy-to-use English Language Command Structure.

FLEX and Color FLEX $179.95

For the "**Strictly Color Computer**"Users

AVAILABLE NOW!!!

Hoyt Sterns Electronics—

Intrigued by **Forth**???? Here is a Forth package tailored to the color computer! This package is supplied on Tape, with instructions for transferring it to disk if you wish. Written primarily in machine language, it's speed is unparalelled. A full Semigraphic-8 Editor is provided, along with "goodies" like Graphics and Sound Commands, Printer Commands, Auto-Repeat and Control Keys, etc. If you are interested in Learning Forth, a Trace Feature is provided which is invaluable. If you are a FORTH Pro, this package provides CPU carry Flag accessibility, Fast Task Multiplexing; Clean Interrupt Handling, etc. (Or; you won't "out grow" the Basic capabilities of the Implementation). Combine this package with Leo Brodie's EXCELLENT Book "Starting FORTH", and you will be a FORTH Expert before you know it (and have a lot of fun doing it!).

$58.95

Introl

AVAILABLE NOW!!!

6809 "C" Compiler; generates extremely efficient object code. Output "benchmarks" close to 10MHz 68000 in 8 Bit Operations; 1.5 times faster than a 4 MHz Z80 when using a 2MHz 6809 System (Re. p 43, '68' Micro Journal, May '83).

FLEX, Color FLEX, OS-9 $375.00 UNIFLEX $425.00
One Year Maintenance. $100.00

For Ordering Call TOLL FREE 1-800-338-6800

MasterCard VISA

SOUTH EAST MEDIA
5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

** QUALITY SOFTWARE NEEDED **

Standard S50 Bus and Color Computer

For the past few months we at the South East Media Division of Computer Publishing, Inc. (CPI), the parent company of 68 MICRO JOURNAL, have debated expanding into the software distribution business. Many other magazines have been doing so for years. Presently there are many fine examples of software that has been developed by YOU our readers, that will never see the 'light of day' unless someone, with enough exposure and willingness to continually advertise, runs with the ball.

Software is the 'backbone' for the **real** utilization of any computer, ours are no exceptions! Realizing that there will be some conflicts, with other advertisiers, this has been no simple decision. However, since day one the foremost concern of 68 MICRO JOURNAL has been it's **readers!** Therefore, South East Media Division will accept, for appraisal, software that runs on 6809 systems, games, utility or applications programs.

In the past there has been too much software offered that was not quite ready, nearly, but not quite. We will strive to eliminate that element. But right up front we tell you only that we will do our very best, nothing more. Also we will strive to keep cost to a bare minimum, while securing for the author a fair return, in royalty payments, promptly paid.

Of course we will expect, no — demand, that the author keep the product free of errors (bugs), and maintain it on a prompt and business like basis. Also we shall require that authors be willing to furnish 'source' for those programs that justify, by price and utility, inclusion of same. The lack of source code, properly commented, is a continual complaint we hear. Not all programs will be sold with source, but where necessary, we will insist that it be included.

In some instances the program may be small or short and not justify itself as a 'single' sale product. In this event it will be combined with other like programs, and offered as a package. In that event the royalties will be split between the various authors.

If you have software that you feel will qualify under this program please contact the proper person as shown below.

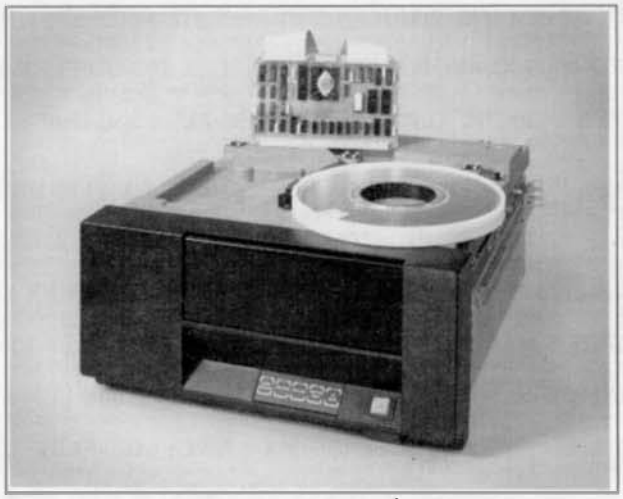
Color Computer	Standard S50 Bus
Tom Williams	Don Williams
Bob Nay	Bob Nay

Remember, if your software has any problems or 'funnies' — GET IT STRAIGHT BEFORE YOU CONTACT US!!!! Also get your source code in proper shape and well commented. There is too much 99% code already drifting around.

South East Media, POB 794, Hixson, TN 37343 — (615) 842-4601
A Division of CPI

MAG Tape Drive/Controller for SS-50 Bus IBM-Compatible

Can't decide? If your tape drive question is "to buy or not to buy," your one answer is SOFTWARE CONSULTANTS. We've got a super IBM-compatible tape drive/controller from the leading manufacturer...and you can buy one directly from us, or, we'll be your service bureau and do your dumping/transferring for you. Either way, you'll get a great deal.



GREAT HARDWARE

Useful

- Allows two way data transfer to and from your system to the big minis and mainframes.
- Software drivers run under OS9 Levels I and II.
- Mag tape device that's usable for hard disk backup under OS9 and as 45 MB of sequential access mass storage.

Powerful

- Usable with any SS-50 bus computer.
- Reads & writes industry standard 1600 bpi phase encoded tape.
- Controller card features onboard microcomputer with 8K buffer.
- Standard drive is Cipher Microstreamer. Others can also be used.

GREAT SERVICE

Fast

- Normally in and out within a week.
- Quick turnaround service (1 day) for rush jobs.

Reasonable

- Our low rates will surprise you. Call and find out.
- For dealers too. Let us get you out of a jam.

SOFTWARE CONSULTANTS

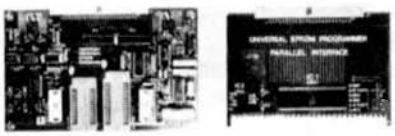
Basic system price $6800.

Phone us with your problem and we'll get down to business.

6435 Summer Avenue • Memphis, TN • 38134 • 901/377-3503

WINDRUSH MICRO SYSTEMS

ALL-IN-TWO EPROM PROGRAMMER



- Probably the most versatile EPROM PROGRAMMER available. Interfaces & software for EXORciser-II (fully addressable) and SS-30 bus systems.
- PROGRAMMES AND VERIFIES 2508/2708, 2516/2716 (SINGLE AND TRI-Volt TYPES) 2532, 2732, 2732A, 2564, 2764 AND THE 128K TMS2528 (16K x 8) -> -> -> WITHOUT ADDITIONAL 'PERSONALITY' MODULES <- <- <-
- PROGRAMMER extends out to your work area via 5' of twisted pair cable.
- EXTENSIVE COMMAND$ MENU.....MOVE DATA, READ, PROGRAM, VERIFY EPROMS, EXAMINE/CHANGE BUFFER, FORMATTED DUMP OF BUFFER, FILL BUFFER.
- Fully documented user's manual W/schematics & theory of operation. Professionally finished PCB's W/solder resist & component overlay
- Software drivers available for (FLEX 2/9), (SSB), (OS-9), and (MDOS). ALL SOURCE FILES SUPPLIED. Specify OP/SYS and disk size on order!
- Binary file READ/WRITE utilities supplied with OS-9 version. Binary file offset loader supplied with MDOS version.
- FULLY ASSEMBLED, BURNED-IN, AND TESTED.....NO EXTRAS TO BUY!

MACE A co-resident EDITOR/ASSEMBLER written by Graham Trott which takes most of the pain out of assembly language program development. Allows programs to be written, edited, assembled, and de-bugged WITHOUT ever entering the disk operating system. Includes XMACE a co-resident 6800/1/3 EDITOR/CROSS-ASSEMBLER.

PL/9 A co-resident EDITOR/COMPILER/DE-BUGGER written by Graham Trott. A single pass compiler that produces position independent machine code output. Supports many BASIC, SPL/M, and PASCAL structures. Supports 8 bit and 16 bit signed AND 32 bit floating point variables. FLEX I/O, floating point, and scientific functions library (w/source) included.

DETAILED OVERVIEWS OF THE ABOVE PRODUCTS ARE ON PAGES 35/36 OF THE OCTOBER 1982 ISSUE OF '68 MICRO JOURNAL.

'C' The FLEX version of the James McCosh 'C' compiler that was originally developed for UNIFLEX. Supports all 'C' data types except 'floats', 'doubles', and 'bit-fields'. Produces very efficient assembly language source output. The TSC relocating assembler/linking loader (SP09-17) is recommended if you wish to make maximum use of C's ability to produce library modules.

MACE (includes XMACE) .....(6809 FLEX ONLY).....	$ 98.00
PL/9 (includes MATHS package) .(6809 FLEX ONLY).....	$198.00
'C' (A 56K 6809 FLEX system is required).....	$295.00
S-30 ALL-IN-TWO, W/one version of software drivers.....	$375.00
EXORciser ALL-IN-TWO, W/one version of software drivers.....	$395.00
SOFTWARE DRIVERS for a 2nd, 3rd or 4th OP/SYS.....	$ 25.00

PRICES INCLUDE AIR MAIL POSTAGE

AN S-30 IEEE-488 TALKER/LISTENER/CONTROLLER WILL BE AVAILABLE SOON!

WE ARE A STOCKING DISTRIBUTOR OF SSB, GIMIX, TSC & MICROWARE. GIMIX IS THE US/CAN. DISTRIBUTOR FOR WINDRUSH.

WORSTEAD LABORATORIES
NORTH WALSHAM, NORFOLK
ENGLAND NR28 9SA
TEL: (0692) 405189
TLX: 97360 SHARET G

*** ALFORD'S GOOD NEWS ***

SALE SALE SALE

PRE-INVENTORY CLEARANCE SALE!!

SALE SALE SALE

Its inventory time around here. To make things easier on ourselves, we want to sell as much of our current inventory as we can. So until stock is depleted, we are offering some FANTASTIC DEALS on selected items. Remember, these prices are ONLY good until our current stock is gone, so call or write to place your order today!

VS-1 SPEECH SYNTHESIZER.....$120

This is the most popular unlimited speech synthesizer available for the SS-50 I/O bus! All units are completely assembled and tested, LESS the SC-01A synthesizer I.C. If you buy this unit while on sale, we will include a copy of our SC-1 text translator program at no additional charge. This gives you a $170 value. You save $50!! Be sure to specify 6800 FLEX , 6809 FLEX , OS-9 , DOS68 or DOS69 operating system and disk size when ordering.

SP-1 SPEECH SYNTHESIZER.....$110

Our popular unlimited speech synthesizer for the Radio Shack Color Computer is also available on this sale. All units are completely assembled and tested, LESS the SC-01A synthesizer I.C. Buy your SP-1 on this sale and we will INCLUDE our SC-1/COLOR translator program which lets your CoCo speak English using nothing more than PRINT statements! Also, we will throw in our COLOR-TREK+ program. This is our COLOR-TREK program adapted for the SP-1. English messages are continually spoken during game play! The action is non-stop. This makes a $170+ value. You save 50+! Software is available on CASSETTE ONLY.

SC-01A SYNTHESIZER IC'S.....$ 55

These chips are the ones used in our synthesizer boards, and are prime, new parts directly from Votrax. We have a limited number of this most popular speech synthesizer I.C. in stock for this sale. Suggested retail price is $85. Discount houses are selling them for $70. You save $15 on the discount house price. We include copies of the Votrax spec sheets with these parts.

SCREDITOR-III.....$150

All of our customers who have used both of the other major word processing packages available for 68xx systems tell us that SCREDITOR III is the best program that they have ever seen (in truth, ONE customer liked one of the others better, but you can't please EVERYONE!). As of July 15th, SCREDITOR III will cost $175. If you have been considering a word processor, or just need a better editor than what you are using now, this is the time to buy SCREDITOR III! For more information about this fantastic program which includes over 150 pages of documentation (and another 240 psges coming soon!), see our ads in the past several issues of Micro-Journal. Be sure to specify 6800 FLEX , 6809 FLEX , OS-9 , DOS68 or DOS69 operating systems and disk size when ordering.

ORDERING INFORMATION...

All sales are FINAL. No returns or refunds, NO open account orders and NO DEALER DISCOUNTS on VS-1, SP-1 or SC-01A purchases for the duration of this sale. Normal dealer discounts and open account terms will apply to SCREDITOR III purchases.

All overseas orders MUST include $10 for shipping. All North American orders under $100 MUST include $5 for shipping. All checks MUST be drawn in U.S. dollars on accounts in U.S. banks. No foreign banks or currencies will be accepted under ANY conditions. All Virginia orders MUST include 4% state sales tax. COD orders in the U.S. ONLY, and must be shipped by UPS. All overseas orders shipped by AIR PARCEL POST. We gladly accept VISA and MASTERCARD orders.

Please allow two to three weeks for order processing. Orders paid by check may take longer. Stocks are limited. Any orders received after stocks are exausted will be returned unprocessed.

ALFORD AND ASSOCIATES
PO BOX 6683
RICHMOND VA 23230
PH 804-320-6722

FLEX is a trademark of Technical Systems Consultants. OS-9 is a trademark of Microware Systems Corporation.

The Lord is our primary associate. Too often with the pressures of business today, time is not taken to acknowladge His presence, and too often we fail, due to our human weaknesses, to show His influence in our lives. We thank each of our customers for their patience and prayers for us when we do not reflect Christ in our business activities.

TEN MOST-ASKED QUESTIONS

ABOUT DYNACALC™

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. **What is an electronic spread-sheet, anyway?**
   Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. **Is DYNACALC just for accountants, then?**
   Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. There's even a built-in sort command, so you could use DYNACALC to manage small data bases - up to 256 records.

3. **What will DYNACALC do for ME?**
   That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalc, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications.

4. **Do I have to learn computer programming?**
   NO! DYNACALC is designed to be used by non-programmers, but even a Ph.D. in Computer Science can understand it. Built-in HELP messages are provided for quick reference to operating instructions.

5. **Do I have to modify my system to use DYNACALC?**
   Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

6. **Will DYNACALC read my existing data files?**
   You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, RMS data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. **How fast is DYNACALC?**
   Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. **Is there a version of DYNACALC for MY system?**
   Probably. You need a 6809 computer (32k minimum) with FLEX or UniFLEX operating system. A version for OS-9 is also in the works. You also need a decent CRT terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX version of DYNACALC also allows you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX and Data-Comp's F-MATE. A version for Frank Hogg's Color Computer FLEX is also being done.

9. **How much does DYNACALC cost?**
   The FLEX versions are just $200 per copy; UniFLEX version $395. Foreign orders add $10 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. **Where do I order DYNACALC?**
    See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 a.m. to 6 p.m., Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX versions. Software serial number is required for the UniFLEX version of DYNACALC.

ORDER YOUR DYNACALC™ TODAY

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 7A Burton St., Darlinghurst, NSW 2010 Sydney. Telephone: 02-357-5111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

CSC

Computer Systems Center

13461 Olive Blvd.

Chesterfield, MO 63017

(314) 576-5020

OS-9 Version NOW Available

$250.00

UniFLEX software prices include maintenance for the first year.

DYNACALC, DYNAMITE, and DYNAMITE + are trademarks of Computer Systems Center.

F-MATE is a trademark of Data-Comp.
VisiCalc is a trademark of VisiCorp.
OS-9 is a trademark of Microware and Motorola.
FLEX and UniFLEX are trademarks of TSC.

ALSO FROM CSC

DYNAMITE +

"THE CODE BUSTER"

now available for UniFLEX
OS-9 version soon

DYNAMITE + is a new version of DYNAMITE, our popular 6809/6800 disassembler package for 6809 FLEX. Present users of DYNAMITE can upgrade to DYNAMITE + by sending us the original DYNAMITE diskette and $40 (plus $5 for foreign postage). DYNAMITE + does everything DYNAMITE does, and more! A cross-reference generator has been added, label files are now maintained only in text form (LABEL EQU $xxxx), and boundary file specifications have been tremendously simplified, which makes it easier to disassemble large programs containing lots of big tables.

The UniFLEX version of DYNAMITE + does everything the FLEX version does, and also automatically handles system calls and 'info' areas.

DYNAMITE + is available for $100 per copy on FLEX (specify diskette size), and $300 on UniFLEX. Foreign orders add $5 per copy for postage.

'68' MICRO JOURNAL

★ The only ALL 6800 Computer Magazine.

★ More 6800 material than all the others combined:

MAGAZINE COMPARISON
(2 years)
Monthly Averages
6800 Articles

KB	BYTE	CC	DOBB'S	TOTAL PAGES
7.8	6.4	2.7	2.2	19.1 ea. mo.

Average cost for all four each month: $6.53
(Based on advertised 1-year subscription price)

'68' cost per month: $2.04

That's Right! Much, Much More
for About
1/3 the Cost!

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Master Charge ☐ — VISA ☐

Card # ____________ Exp. Date ____________

For ☐ 1-Year ☐ 2 Years ☐ 3 Years

Enclosed: $____________

Name ____________

Street ____________

City ____________ State ____________ Zip ____________

My Computer Is: ____________

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

SUBSCRIPTION RATES

USA
1 Year $24.50, 2 Year $42.50, 3 Year $64.50

*FOREIGN SURFACE Add $12.00 per Year to USA Price

*FOREIGN AIRMAIL Add $36.00 per Year to USA Price

**CANADA & MEXICO Add $5.50 per Year to USA Price
Cash (USA) or drawn on a USA Bank!!!

OMEGASOFT INDUSTRIAL STRENGTH PASCAL

OMEGASOFT PASCAL

If you're looking for a language to write real-time process control software, look no further. With the rising cost of labor, it is becoming more critical that a high level language be used. Find out why over 500 companies have switched to OmegaSoft Pascal for their demanding applications.

WHY PASCAL?

Pascal was designed to teach students how to write structured programs that are easy to read and maintain. In the past decade it has also proved to provide the same advantages in industrial applications.

EXTENSIONS

OmegaSoft has taken the Pascal framework and expanded the basic data types, operators, functions, and memory allocation to fit the needs of real-time systems. These additions fit in the same structure as Pascal and enhance its usefulness without impairing the excellent readability, ease of maintenance, and structured design.

The byte data type allows you to directly address bytes in memory or I/O devices. The common arithmetic operations can be used for bytes along with shift left, shift right, "and", "or", "eor", and complement operators. These operators are also available for integer and hex (2 byte unsigned) numbers.

Longintegers are four byte signed numbers useful for extended range arithmetic commonly needed for machine control. Functions have been added to allow conversion between the various data types. Dynamic length strings allow complex text manipulation and allow effective interactive I/O.

Variables can be placed either on the data stack (default), at an absolute address in memory (for I/O), in base page, relative to the program counter (for constant tables), or defined in another module.

FEATURES

The compiler generates assembly language for assembly and linking to run on the target system. Since a true relocating assembler and linking loader is used, only those runtime modules required are automatically linked in, providing a much smaller object module than other compilers.

Large Pascal programs can be split up into conveniently sized modules to speed the development process. Procedures, functions, and variables can be referenced between Pascal modules and assembly language modules by using Pascal directives.

Full source code is included for the runtime library, the debugger, and other support utilities.

ISO COMPATIBILITY

OmegaSoft Pascal has been tested using the pascal Validation Suite. The Suite is a collection of over 400 Pascal programs designed to test the quality of Pascal Compilers and their runtime systems for compliance with the ISO (International Standards Organization) Pascal standard. OmegaSoft is the only supplier of 6809 native Pascal compilers that publishes this report in its instruction manual.

DEBUGGER

The compiler package includes an interactive, symbolic debugger. The debugger allows setting of breakpoints, displaying and changing variables, and tracing statements. The debugger allows very fast turnaround for programs to be run on the host system.

TARGET SYSTEMS

The target system may be any 6809 system. No specific I/O devices are required. The output code is re-entrant and rom-able, perfect for single-board systems up to large development systems. There are no charges for use of the output of the compiler or the object of the runtime library in your products. 68000 target coming in 3rd Qtr. 1983.

HOST SYSTEM

The host system must be 6809 based and have at least 48K of ram (56K recommended) and run one of the following operations systems: MDOS, XDOS, OS-9, or FLEX. Priced from $425. 68000 host coming in 4th Qtr. 1983.

SUPPORT PRODUCTS

The OmegaSoft Relocatable Assembler and Linking Loader is designed to support the Pascal Compiler Package and can also be used for general assembly language program development. Priced from $125.

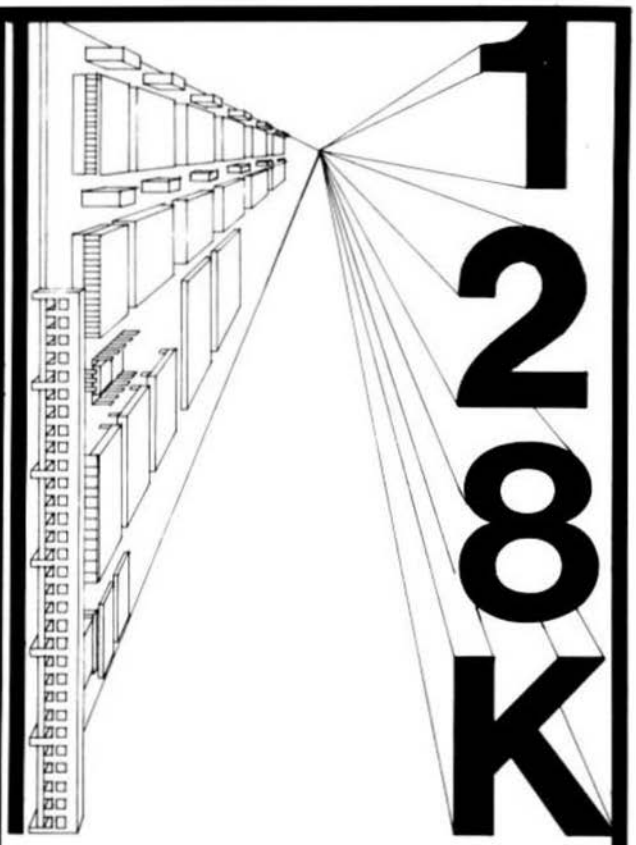
OmegaSoft's Screen Editor supports smart terminals and comes complete with the Pascal source. Priced from $90.

Faster floating point execution can be obtained by use of the Arithmetic Processor Unit option which uses the AMD9511 APU chip. Priced from $90.

For complex real-time applications, the Multi-tasking Kernel provides task scheduling, inter-task communications, and resource interlocking. The Kernel is a runtime library that is accessible as Pascal functions and procedures (with full source included). Priced from $175.

Dealer and OEM inquiries invited. OmegaSoft products are also available from distributors in Australia and Western Europe, call or write for more information.

OMEGASOFT
P.O. Box 842
Camarillo, CA 93010
(805) 987-6426



Print Buffer

- Holds over 50 average-size pages
- On board firmware with many software features
- RS-232-C serial or centronics type parallel versions available
- MC6809 on board microprocessor
- Plugs directly into SS-30 and SS-30C computers (SWTPc, GIMIX, SSB and others)
- Tested, complete with owner's manual

VCM-128P: Parallel Version	$459.00
VCM-128S: Serial Version	$469.00
Cable Assembly (Serial or Parallel)	$ 19.95

Be Sure To Ask About Our 8K Buffer/Interfaces

Please add $5.00 for USA orders, $15.00 for overseas shipping/handling.

MC/VISA
Dealer Inquiries Welcome

ChiraTech Scientific Instruments

204 N. Link Lane, Alpha 9
Fort Collins, Colorado 80524
(303) 484-1913

ARCADE 50

POWERFUL COLOR GRAPHICS
Uses the new TMS9918A Video Display procesor. High resolution 256 x 192 pixel display with 15 colors. 16K Bytes of onboard RAM does not reduce user memory. 32 graphic images can be individually moved with simple X-Y commands for smooth animation. External Video input allows subtitling. NTSC composite video output.

SOUND EFFECTS AND MUSIC
- Three AY3-8 10 Programmable Sound Generators
- Nine simultaneous voices
- Three independent noise sources
- Onboard stereo amplifier drives two 8 ohm speakers

ADDITIONAL I/O CAPABILITIES
- Eight analog inputs with 8 bit resolution
- Supports four joysticks with pushbutton switches
- Eight bit parallel I/O port
- Entire unit maps into 256 bytes of memory

FBASIC

TERMINUS DESIGN INC, in conjunction with Microware Systems Corporation, is proud to announce FBASIC an enhancement of Microware's 6800/BASIC. Their fast compiled BASIC has been adapted for 6809 users with added video and sound features for ARCADE 50 users. FBASIC is a true compiler that produces optimized machine language modules which are ROMable and require no Run-Time package. FBASIC requires less memory overhead and runs hundreds of times faster than BASIC interpreters. It supports standard BASIC instruction including String functions. Disk I/O and fast integer arithmetic with multiple-precision capability. Graphics verbs and functions fully support the Arcade 50.

ARCADE 50 assembled and tested	$325.00
Video and Audio connector set	15.00
4 Joystick connector set	15.00
2 Radio Shack Joysticks	24.00
Gold Molex connectors	12.00
A/BASIC for 6800	110.00
FBASIC for 6809	110.00
FBASIC (with ARCADE 50)	75.00
ARCADE 50 RGB	375.00
LABVIDEO (Motorola EXORbus)	375.00
NEW MVB 6809 Processor Board	225.00
256K Dynamic Memory Board	795.00
256K Dynamic Memory Board (w/64K)	395.00
64K Dynamic Memory Board	295.00

TERMINUS DESIGN INC
16 SCARBROUGH ROAD
ELLENWOOD, GA 30049
(404) 474-4866

68 MICRO JOURNAL PROGRAMS on DISK

Disk #1: FILESORT, MINICAT, MINICOPY, MINIFMS, **LIFETIME, **POETRY, **FOODLIST, **DIET.
Disk #2: DISKEDIT w/ Inst. & fixes, PRIME, *PRMOD, **SNOOPY, **FOOTBALL, **HEXPAWN, **LIFETIME.
Disk #3: CBUG09, SEC1, SEC2, FIND, TABLE2, INTEXT, DISK-EXP, *DISKSAVE.
Disk #4: MAILING PROGRAM, *FINDDAT, *CHANGE, *TESTDISK.
Disk #5: *DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING.
Disk #6: **PURCHASE ORDER, INDEX (Disk file Indx).
Disk #7: Linking Loader & RLOAD, Harkness
Disk #8: CRTSET, Lanpher (May '82)
Disk #9: DATECOPY, DISKFIX9 (Aug '82)

NOTE: All are as published or received by 68 Micro Journal, some have fixes and patches.

This is a reader service only! No Warranty is offered or implied, they are as received and are for reader convenience ONLY. Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other.

PRICE: 8" Disk $19.95 - 5" Disk $17.95

68 MICRO JOURNAL
POB 794
Hixson, TN 37343
615-842-4600

* indicates 6800, ** indicates BASIC SWTPC or TSC - 6809 no indicator.

MASTER CARD - VISA accepted - Foreign add sufficient postage surface or air!!

64K SS-50 STATIC RAM

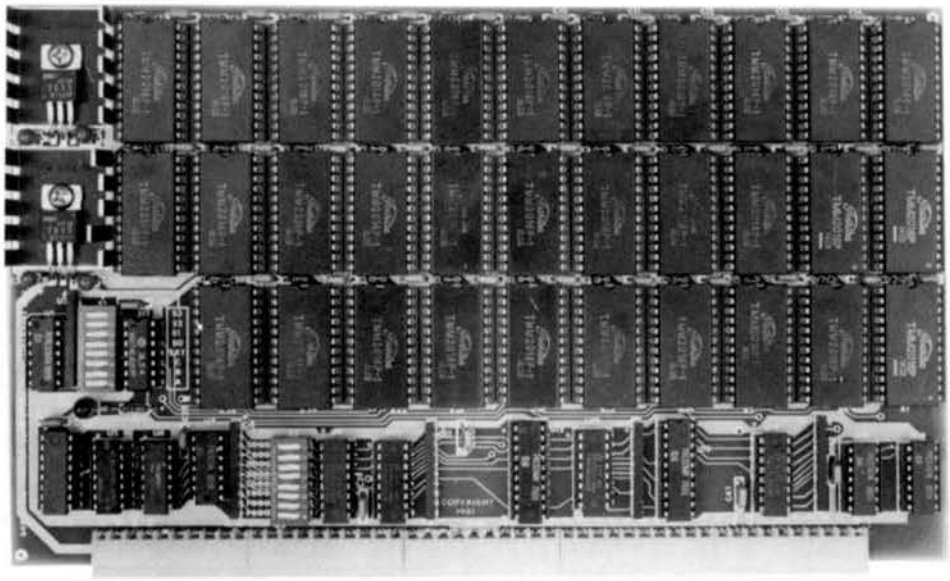
$179.00 (48K KIT)

NEW!

NEW!

LOW POWER!

RAM OR EPROM!



BLANK PC BOARD WITH DOCUMENTATION $52

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $50**

FEATURES:

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

56K	$219
64K	$249

CLOSE OUT SPECIAL
WE HAVE DROPPED OUR 32K SS-50 STATIC RAM BOARD WHICH USED 2114 LOW POWER RAMS. WE WILL SELL THE REMAINING STOCK OF BLANK PCB'S WITH DATA FOR $17.50 EA. THESE FORMERLY SOLD FOR $50.

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50, add 85¢ for insurance.

THE BEST JUST GOT BETTER *SUMMER SPECIALS* FHL FLEX ONLY $69.95

FHL CC-FLEX has become the standard FLEX for the CoCo. Now you can buy CC-FLEX for only $69.95 for a savings of $30. CC-FLEX includes more extras, more utilities, and more functions than any other.

INCLUDES:

- **BEST PRICE IN THE WORLD!**
  At $69.95, CC-FLEX, with all these features, is the best price anywhere!
- **OFFICIALLY LICENSED TO FHL FROM TSC**
  CC-FLEX is an official licensed FLEX from TSC. Beware of unlicensed implementations.
- **NEW SMOOTH SCROLLING**
  New with version 5.0.4 is SMOOTH scrolling. This is a feature not found on any other FLEX.
- **NEW VARIABLE RATE SCROLLING**
  Variable rate scrolling lets you control the speed that the screen scrolls, from smooth to very fast scroll. Also this version of CC-FLEX is 50% faster in scrolling than previous versions.
- **NEW 'TED' TINY EDITOR**
  Included with CC-FLEX at $69.95 is 'TED', which stands for Tiny Editor. TED is fine for small editing jobs and is the easiest editor to use of any available. You can learn how to use it in less than five minutes!
- **NEW 'ISM' INTERACTIVE ASSEMBLER**
  ISM is the program for those interested in assembly language but not convinced that they need to buy one. ISM is the perfect teaching tool to use to learn assembly language programming. You will need to use it in conjunction with a book on the subject (not provided). However, ISM includes a complete manual on its operation.
- **NEW EXTERNAL TERMINAL PROGRAM**
  If you want to run CC-FLEX connected to a terminal so that you can get the benefit of a real keyboard and full 24 x 80 display, then EXT will do it for you. EXT allows a standard terminal and printer to be connected to the CoCo. 'NEW' EXT now supports real hardware handshaking.
- **EASY START UP**
  Just type RUN"FLEX" or DOS if you have 1.1 Disk ROM. Quote from the June 1983 issue Hot CoCo. Re: FLEX by David Wasler, pg. 143, "It is the easiest to use. After you receive it, just put it in your drive and type RUN"FLEX"."
- **ONLINE "HELP" CAPABILITY**
  Just like the big mainframes, CC-FLEX has a help function. Just type 'HELP' and CC-FLEX will answer your questions and help you run FLEX without having to keep paging through the manual.
- **RECONFIGURABLE TO YOUR NEEDS**
  CC—FLEX'S powerful SETUP facility lets you change CC-FLEX to suit your needs and hardware. CC-FLEX is the only FLEX that gives you such complete control.
- **SUPPORTS ALL DRIVE TYPES AND SIZES, EVEN 3"**
  CC-FLEX supports every type of drive currently available for the CoCo. These include 35, 40 and 80 track single and double sided. Even the new 3 inch from AMDEX. We supply support software on the 3 inch drive.
- **SIX DIFFERENT HI-RES SCREENS**
  51 x 24 and 16 x 32 with true lower case, 64 x 24 and 64 x 32 all uppercase. You can switch between these screens at will. The different scroll types are available for each screen type.
- **SUPPORTED BY THE LARGEST 6809 SOFTWARE FIRM IN THE WORLD!**
  FHL is the largest software house in the world for 6809 based computers! We have over 100 software packages for FLEX and OS-9. We support CC-FLEX like no other.

OPTIONS AVAILABLE AT EXTRA COST:

- **DBASIC** - Radio Shack Disk Basic adapted to work with FLEX.
  This is standard RS Disk Basic working with FLEX. This only works with FHL FLEX at this time. DBASIC has everything that RS Disk Basic has except for Direct access files and those functions that go with Direct access files.
- **ED/ASM** - Full feature Editor and Assembler
  SPECIAL now only $69.95! ED/ASM is the best Editor and assembler for the Color Computer. The editor is both screen and line oriented while the assembler is a full conditional macro assembler.

REVIEWS:

Quote from the June 1983 issue of HOT COCO. Re: FLEX by David Wasler, pg. 143.
"Frank Hogg Color FLEX has been on the market the longest and has the most software support. It is also the easiest to use. After you receive it, just put it in your drive and type RUN"FLEX"."

Quote from the March 1983 80 MICRO review by Scott Norman, pg. 101.
"I think CC-FLEX offers the most painless way of trying one's hand at an advanced operating system for the Color Computer. It offers quite a few "big machine" features, and opens the door for a lot of applications software."

Frank Hogg Lab brought FLEX to the CoCo over a year and a half ago. It has gone thru 5 updates since its original release in February 1982! FHL not only has FLEX, we support our FLEX with more software than anyone else in the world! We have over 100 software packages compatible with FHL CC-FLEX to fulfill all your needs. These range from languages, utilities and software development tools to data base management, word processing and business applications. We guarantee these programs will work with our FHL CC-FLEX and probably with any licensed version of FLEX!

And, there is no better way to purchase FLEX for less!!

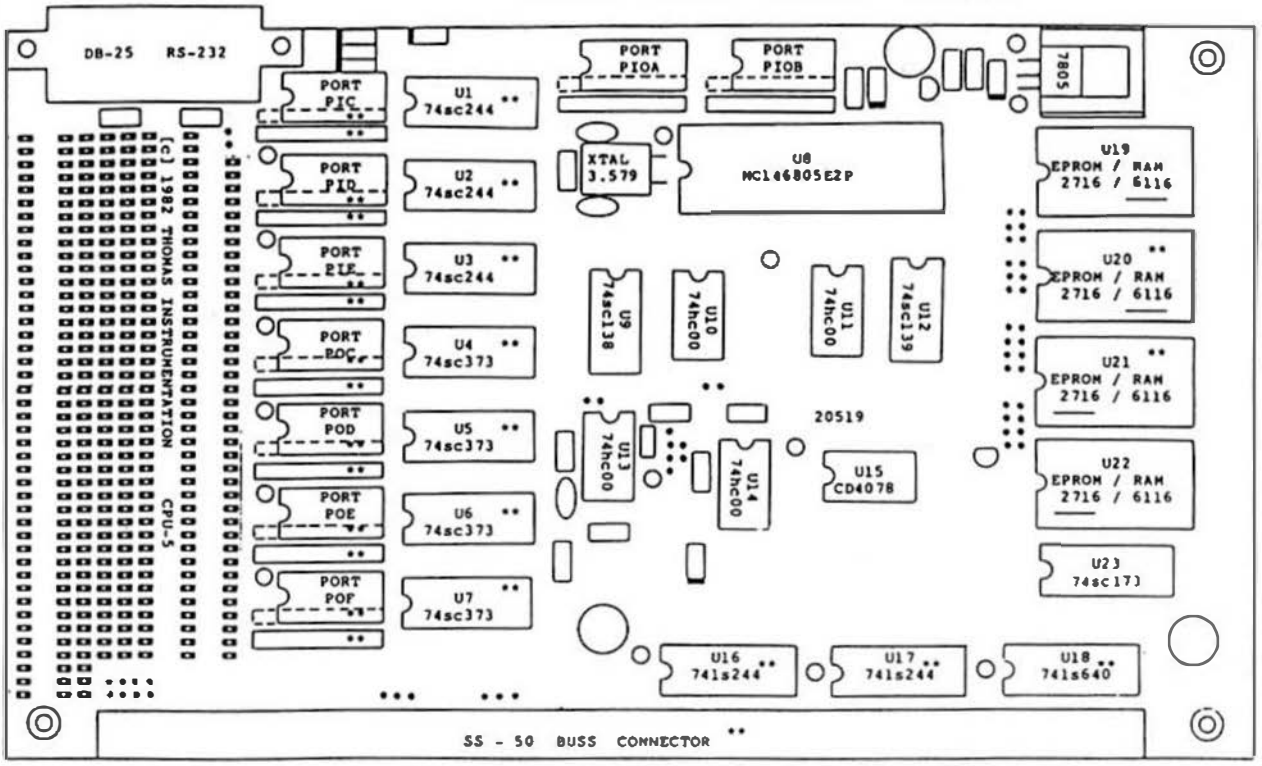
So, order FHL CC-FLEX today and take advantage of our **SUMMER SPECIAL of $69.95** to bring out the true power of your CoCo!!

We will accept Prepaid, COD, VISA, M/C and Diners. Please include $3.50 for shipping and handling.

FHL FRANK HOGG LABORATORY

THE REGENCY TOWER • 770 JAMES ST. • SYRACUSE, NY 13203 • TELEX 646740 • (315) 474-7856

*** NEW 6805 ALL CMOS CARD ***



CPU-5-B	Blank card with documentation	$49.00	
CPU-5-P	Partial card (parts marked ** not incl.)	$229.00	Includes EPROM Monitor and 115 vac Power Pak.
CPU-5-F	Full card with all parts shown (fully assembled)	$349.00	
CROSS-ASSEMBLERS	for FLEX ([c] TSC) 6809 system	$150.00	

SOFTWARE

VDISK Treat extended memory as a super fast disk drive

6809 source & object	$149.00
6809 object	$ 99.00

OUTSIDE MODEM PROGRAM incl. source

UniFLEX version	$100.00
FLEX version 6800 & 6809	$ 50.00
HAYES SMART MODEM	$249.00

CROSS ASSEMBLERS for 6800,6801,6805 runs on 6809 FLEX $150.00

TV-EDIT Screen oriented Editor for FLEX 6809 $ 95.00
Please specify 5" or 8" Disk when placing order.

CHIPS

Please note when ordering IC's that a $100.00 Minimum is in effect Orders less than the minimum will be charged an additional $10.00 for handling

6116P-3	2MHZ	$13.50
2016P-2	2MHZ	$10.50
2716	1MHZ	$ 6.50
2716-1	2MHZ	$16.50
146805E2P		$25.00
68B02		$10.00
68B09		$25.00
68B21		$ 6.50
68B40		$12.00
68B50		$ 6.75
74LS640		$ 3.25

HARDWARE

NEW MEMORY PRICES TO INTRODUCE VDISK

S-R/R without memory chips	$120.00
with 8K NMOS@2MHZ	$169.00
with 16K NMOS@2MHZ	$199.00
with 32K NMOS@2MHZ	$299.00
with 48K NMOS@2MHZ	$399.00
Bare Card	$ 49.00

Extender Cards assemb with logic aid

SS-50/50C	$ 35.00
SS-30	$ 25.00
SP-1 Prototype Board A/T	$195.00
SP-1 Bare Card	$ 49.00
SS-50 Wire Wrap Board B/C	$ 39.00
SS-30 Wire Wrap Board B/C	$ 20.00
6802 Super CPU A/T	$235.00
6802 Super CPU B/C	$ 59.00
Video RAM, B/C $49.00, A/T	$195.00
Parallel I/O, B/C $49.00, A/T	$139.00
SS-50 Backplanes 4,6,8,12,&16 position @ $5.00 per slot w/o connectors	
SS-30 8 pos. BP w/o connect.	$ 39.00
Transition Card $49.00, A/T	$ 95.00
Molex Gold Male $1.60, Female	$ 1.60
Molex Tin Male $ .40, Female	$ .50

THOMAS INSTRUMENTATION

168 EIGHTH STREET — AVALON, N.J. 08202 (609) 967-4280
NJ RES. INCLUDE 5% SALES TAX
Allow 3 Weeks for Checks to Clear
CONT. USA INCLUDE $3.00 SHIPPING, CANADA $6.00, FOREIGN $12.00
MASTERCARD, VISA, and C.O.D. ACCEPTED
Send $20.00 to receive full documentation, schematics, & source listings for all boards currently in production.

'68' MICRO JOURNAL ADVERTISERS INDEX

This Index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

TO REALIZE THE FULL POWER & PERFORMANCE OF THE 6809, LOOK TO GIMIX.

GIMIX OFFERS YOU A VARIETY OF SS50 BUS COMPONENTS AND SYSTEMS.

OS-9 GMX III

The GMX 6809 CPU III and OS-9 GMX III. A Multi-user, Multi-tasking package for the ultimate in System Performance plus protection of the system and other users from crashes caused by errors in individual users programs.

#01 (CPU & Software) ..... NEW! **$1698.01**

INTELLIGENT I/O PROCESSOR BOARDS increase system throughput by reducing interrupts to the host, buffering data transfers, and data preprocessing. Prices include on board firmware. Requires system drivers.

#11 3 port RS232 Serial (SS30) ..... NEW! **$498.11**
#12 4 port Parallel (SS50) **$538.12**
OS-9 GMXIII drivers . . (included when purchased with GMX III package) . . **$200.00**
OS-9 Level 2 users -- contact GIMIX for system requirements and availability.

192K GMX III #79 SYSTEMS: All include GMX 6809 CPU III and OS-9 GMX III (#01); a #11 3 port intelligent serial I/O & cables; #19 Classy Chassis; 192KB Static RAM, #68 DMA controller, all necessary cables, power regulators, and filler plates. The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB are included.

#79 with dual 40 track DSDD drives **$5998.79**
#79 with dual 80 track DSDD drives ..... NEW! **$6298.79**
#79 with **#88** 8" Dual Drive Disk System **$7598.79**
#79 with **#90** 19MB Winchester subsystem & one 80 track DSDD drive . . **$8998.79**
UniFLEX for the GMX 6809 CPU III and Intelligent I/O boards is in development.

OS-9 GMX I; OS-9 GMX II; FLEX; and UniFLEX

The **#05** GIMIX 6809 PLUS CPU board **$578.05**
Options: GMX DAT ..... **$35.00** SWTPC DAT **$15.00**
9511A ..... **$312.00** 9512 ..... **$265.00**

#49 64K8 GHOST SYSTEM includes: #05 CPU; #19 Classy Chassis; 64KB static RAM; a #43 2 port serial card & cables; #68 DMA Controller; all necessary cables, power regulators, and filler plates; GMXBUG monitor; FLEX; and OS-9 GMX I. You can software select either FLEX or OS-9. The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB are also included.

#49 with dual 40 track DSDD drives **$4398.49**
#49 with dual 80 track DSDD drives **$4698.49**
#49 with **#88** 8" Dual Drive Disk System **$5998.49**
#49 with **#90** 19MB Winchester subsystems & one 80 track DSDD drive **$7398.49**

#39 128K8 SYSTEM includes: #05 CPUwDAT; #19 Classy Chassis; 128KB of static RAM; a #43 2 port serial card & cables; #68 DMA Controller; all necessary cables, power regulators, and filler plates; GMXBUG monitor; FLEX; and OS-9 GMX II. You can software select either FLEX or OS-9. The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB, and GMX-VDISK for FLEX are included.

#39 with dual 40 track DSDD drives **$4998.39**
#39 with dual 80 track DSDD drives **$5298.39**
#39 with **#88** 8" Dual Drive Disk System **$6598.39**
#39 with **#90** 19MB Winchester subsystem & one 80 track DSDD drive **$7998.39**
UniFLEX, available at extra cost, requires 8" or Winchester drives. A signed license agreement with TSC is required before shipment.

You can add to any GIMIX system RAM, I/Os and other options, or substitute non-volatile RAM. GIMIX will customize to your needs.

COMING SOON: Contact GIMIX for price and availability on 40MB and 72MB Winchester (5¼") drives, removeable pack Winchesters, 256KB static RAM boards.

All GIMIX systems are guaranteed for 2MHz operation. GIMIX systems include documentation for all boards and software in a GIMIX binder. **ALL DRIVES ARE 100% TESTED AND ALIGNED BY GIMIX.**

ALL BOARDS AND SYSTEMS ARE ASSEMBLED, BURNED-IN, AND TESTED. GOLD-PLATED BUS CONNECTORS ARE USED.

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60690, account #73-32033. Visa or Master Charge also accepted.

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.

GIMIX Inc. reserves the right to change pricing, terms, and product specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Choose from GIMIX' wide variety of system components.

The GIMIX CLASSY CHASSIS #19 consists of a heavyweight aluminum cabinet, constant voltage ferro-resonant power supply, and SS50 Mother board with baud rate generator board $1398.19

Triple Disk regulator card and cables	$88.22	Baud rate generator card	$88.93
Missing cycle detector	$38.23	Filler plates	$14.92
Back panel connector plates (specify)	$8.60	50 Hz. option	$30.00

MEMORIES (GIMIX uses only Static RAM)

#67 64KB NMOS STATIC RAM board **$478.67**
#64 64KB CMOS STATIC RAM board w/battery back-up **$568.64**
#34 8K PROM board **$98.34**
#32 16 socket PROM/ROM/RAM board **$238.32**

I/O Boards (see above for Intelligent I/Os)

#41 Single port serial, RS232/20ma. current loop **$88.41**
#43 2 port serial, RS232 ..... **$128.43**
#46 8 port serial, RS232 ..... **$318.46**
#42 2 port parallel **$88.42**
#45 8 port parallel **$198.45**
#50 serial, RS232, RS422, RS423 ..... **$244.50**
#52 SSDA serial, RS232, RS422, RS423 ..... **$254.52**
#54 ADLC serial, RS232, RS422, RS423 ..... **$268.54**
Each cable with connectors for back panel mounting (specify board) **$24.95**

DISK CONTROLLERS

#68 DMA (featured in all systems above) **$588.68**
#28 dbl. dens. programmed I/O (5" drives only) **$298.28**
#58 single dens. programmed I/O (5" and/or 8" drives) **$226.58**
#48 same as **#58** but for 5" drives only **$198.48**
Cable sets: 8" with Back Panel connector **$29.25**
for two 8" external drives **$44.26**
for two 5" drives **$34.96**

SOFTWARE: GIMIX exclusive versions of OS-9/GMX I, II, III & FLEX are for GIMIX hardware only. All versions of OS-9 require the #68 controller.
When ordered with any controller, FLEX is **$30.00**

GMXBUG PROMs and manual			**$98.65**
Boot or Video boot PROM	**$38.00**	UNIFLEX boot PROM	**$58.00**
OS-9 GMX I	**$208.00**	OS-9 GMX II	**$508.00**
Editor	**$125.00**	Assembler	**$125.00**
BASIC-09	**$200.00**	RUNB	**$108.00**

DISK DRIVES FOR GIMIX SYSTEMS -- complete with cables and power regulators.

5" DSDD 40 track ..... 2 for **$900.00**
5" DSDD 80 track ..... 2 for **$1300.00**
#88 8" Dual 8" DSDD drives, cabinet, power supply, & cables **$2698.88**

Cabinet only	**$848.18**	220V 50Hz. Option, add	**$38.00**
Filler plate	**$14.83**	Cable for 2 drives	**$44.82**
Cable for 4 drives	**$87.84**	Cable for cabinet to mainframe	**$45.81**

WINCHESTER SUBSYSTEMS: for use only in GIMIX systems with #68 DMA controller.

#90: includes one 19MB drive, interface, and Software **$3588.90**
#91: includes two 19MB drives, interface and Software **$5288.91**
Contact GIMIX for price and availability of other forthcoming subsystems.

OTHER BOARDS

#75 GHOST 80X24 VIDEO BOARD **$398.75**

#65 50 pin Protoboards	**$56.86**	**#33** 30 pin Protoboards	**$38.33**
#03 6800 CPU			**$224.03**
#06 6800 CPU with timers	**$288.06**	Baud rate option, add	**$30.00**

#08 RELAY DRIVER (board, bracket, transformer, and 31 relays) **$1128.08**
#05 - #08 (board, bracket, transformer, without relays) **$538.86**
#05 OPTO board **$348.86**
WINDRUSH EPROM PROGRAMMER **$375.00**

3" Binder	**12.00**	2" Binder	**$8.00**

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

DON'T SEE IT??? ASK! OUR BROCHURE HAS MORE COMPLETE DESCRIPTIONS AND SPECS. PHONE OR WRITE TODAY FOR YOUR COPY.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA Inc. FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc. GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

GIMIX INC. 1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

The Original FLEX™ for Color Computers

- Upgrade to 64K
- RS to FLEX, FLEX to RS file transfer ability
- Create your own character set
- Automatic recognition of single or double density and single or doubled sided
- All features available for either single or multiple drive systems
- Settable Disk Drive Seek Rates
- Faster High Resolution Video Display with 5 different formats
- Save RS Basic from RAM to Disk
- Move RS Basic to RAM
- Load and save function on FLEX disk
- 24 Support Commands 12 with Source Text
- External Terminal Program

Languages Available

Pascal, Fortran, RS Basic, RS Assembler, TSC Basic, TSC Assembler, Relocating Assembler, Macro Assembler, Mumps

If you are tired of playing games on your TRS-80C Color Computer, or find that you are handicapped by the limitations of the RS BASIC in trying to write a Program that will allow you to actually USE the Color Computer as a COMPUTER, YOU ARE READY TO MOVE UP TO THE **FLEX9**™ Operating System. If you want to have REAL PROGRAMMING POWER, using an Extremely Powerful Business BASIC, PASCALs, C Compilers, a full-blown Macro Assembler with a Library capability so you are not continuously reinventing the wheel, YOU ARE READY TO MOVE UP TO THE **FLEX9** Operating System. If you would like to see if YOU REALLY COULD USE A COMPUTER IN YOUR BUSINESS, or begin to make your Computer start PAYING IT S OWN WAY by doing some Computer Work for the millions of small businesses around you, such as Wordprocessing, Payroll, Accounting, Inventory, etc., then YOU ARE READY TO MOVE UP TO THE **FLEX9** Operating System. How?? **DATA-COMP** has the way!

DATA-COMP s FLEX9 Conversion for the TRS-80C Color Computer was designed for the SERIOUS COMPUTER **USER**, with features like greatly increased Display Screens, WITH Lower Case Letters, so you can put a FULL Menu on ONE Screen, or see SEVERAL Paragraphs at the same time, with features like providing a FULL Keyboard so you have FULL Control of your Computer AND it s Programs NATURALLY, without needing a chart to see what Key Combination will give you what function, with USER ORIENTED functions to make using the Operating System natural, like having the Computer AUTOMATICALLY determine what type of Disk is being used in what type of Disk Drive and working accordingly, rather that you have to specify each and every thing for it, or like having the Computer work with the Printer you have been using all along without you having to tell the new Operating System what is there, etc.

™FLEX is a trademark of Technical Systems Consultants

DATA-COMP has everything you need to make your TRS-80C Color Computer WORK for YOU: from Parts and Pieces to Full, Ready To Use SYSTEMS. DATA-COMP designs, sells, services, and SUPPORTS Computer SYSTEMS, not just Software. CALL DATA-COMP TODAY to make your Computer WORK FOR YOU!

System Requirements

FLEX9 Special General Version + Editor & Assembler (which normally sell for $50.00 ea.)	$150.00
F-MATE(RS) FLEX9 Conversion Rout. for the RS Disk Controller when purchased with Special General FLEX9 Sys	$49.95
when purchased without the General FLEX9 Sys	$59.95
Set of Eight 64K RAM Chips w Mod. Instructions	$59.95
Color Computer with 64K RAM and EXT. BASIC	$399.95

SPECIAL SYSTEM PACKAGES

64K Radio Shack COLOR COMPUTER, Radio Shack COLOR DISK CONTROLLER, a Disk Drive System, Special General Version of FLEX9™, F-MATE(RS)™ and a Box of 10 Double Density Diskettes; a COMPLETE, ready to run SYSTEM on your Color TV Set. $1079.95

DISK DRIVE PACKAGES, etc.

These Packages **include** the Radio Shack Disk Controller, Disk Drives with Power Supply and Cabinet, and Disk Drive Cable.

PAK #1 — 1 Single Sided, Double Density Sys	$489.95
PAK #2 — 2 Single Sided, Double Density Sys	$749.95
PAK #3 — 1 Double Sided, Double Density Sys	$569.95
PAK #4 — 2 Double Sided, Double Density Sys	$919.95
PAK #5 — 2 Qume Thinline Double Sided Double Density Sys	$749.95

PARTS AND PIECES

Radio Shack Disk Controller	$179.95
1 Single Sided, Double Density Disk Drive, Tandon	$249.95
1 Double Sided, Double Density Disk Drive, Qume	$349.95
1 Qume Thinline Double Sided, Double Density	$279.95
Screen Clean — Clears Up Video Distortion On Your Color Computer	$39.95
Single Drive Cabinet with Power Supply	$89.95
Double Drive Cabinet with Power Supply	$109.95
Single Drive Disk Cable for RS Controller	$24.95
Double Drive Disk Cable for RS Controller	$34.95
Micro Tech Prods., Inc. **LOWER CASE ROM** Adapter	$74.95
Radio Shack BASIC **Version 1.1 ROM**	$34.95
Radio Shack Extended Basic ROM	$89.95

TERM — AN External Terminal Driver is FREE with purchase of F-Mate

LAST OF PRODUCTION!!!

!! LIMITED QUANTITY !!

SWTPC 8212 & 8212(W) Intelligent Terminals

Data-Comp Has Bought the Remaining Supply of SWTPC 8212 CRT Terminals

If You Need an 8212 We Have Them In Stock!

PRINTERS

The Epson MX-80
$425.00
The Epson MX-100
$725.00

MX-70 $355.00 MX-80 FT $575.00

☆

Verbatim Diskettes.

5 Soft Sector Disks

Single Side Single Density	$2.75 ea
Single Side Double Density	$2.75 ea
Double Side Double Density	$4.92 ea
Plastic Storage Box	$2.00 ea

8 Soft Sector Disks

Single Side Single Density	$3.75 ea
Single Side Double Density	$4.10 ea
Double Side Double Density	$4.75 ea
Plastic Library Box	$5.00 ea

Foreign Orders Add 10% Surface, 20% Air Mail

MasterCard VISA

DATA-COMP

P.O. Box 794 *HIXSON, TN 37343*
1-615-842-4601

EM-82 Video Terminal

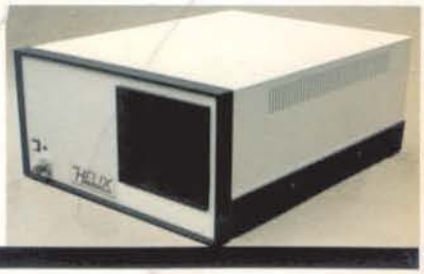
Emulates the 8200 Series from SWTPC

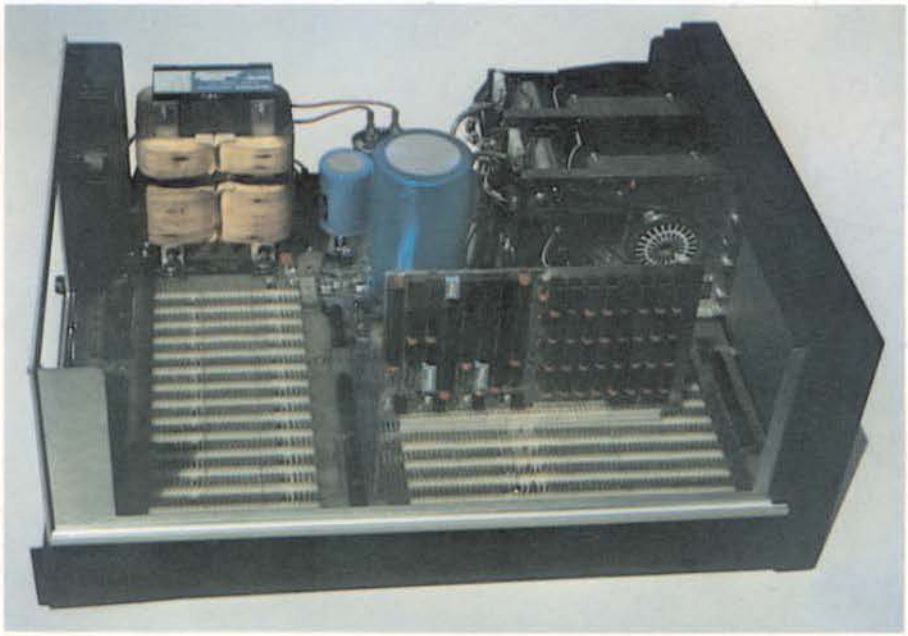
Data-Comp—South East Media & 68 Micro Journal Are Divisions of CPI

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025

HELIX™





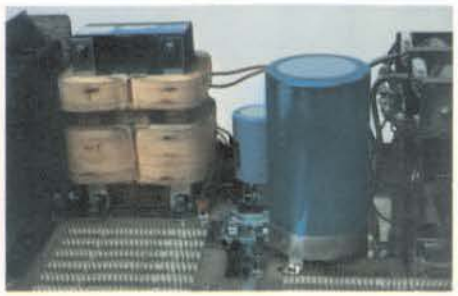
THE MAINFRAME

- Industry Standard Optima™ Cabinet
- Largest Constant Voltage Power Supply in the industry
- S-64 Bus gives 16 Bit Power and S-50 Bus Compatibility
- 10 Main (S-64) Slots
- 14 I/O (S-30) Slots plus 2 On-board
- On-board Baud Rate Generator to 38.4Kb
- Space and Power for two 5¼" Disk Drives
- Full Address Decoding for I/O Slots
- Two RS-232 Serial and Two parallel Ports On-board
- Single Board Construction for Reliability
- Faraday Shielded Bus Lines give "Text Book Clean" Signals

THE PROCESSORS

6809

- Standard 2 MHz Operation
- Standard DAT Compatible with GIMIX and SWTPC
- Standard 6840 Interval Timer
- Standard 1K Scratchpad RAM
- Standard Clock/Calendar with Battery
- Provision for Programmers Console



The HELIX™ computer system represents the latest advance in S-50 bus computer systems. Relying on the physical nature of S-50 bus connectors to guarantee compatibility, the HELIX adds 14 bus lines (becoming S-64) to allow a 68000 processor to operate with full 16 bit data transfer and 24 bit addressing, while at the same time providing full interchangability with existing S-50 components.

Offered with a selection of processors, memories, and peripheral controllers, a HELIX system can be configured for applications ranging from advanced hobbyist to multiterminal time-sharing.

Designed to offer the utmost in speed, reliability, and utility at a reasonable price, it represents a new standard of quality for those who require a professionally designed computer for professional use.

THE POWER SUPPLY

- Ferro-resonant Transformer for Line Noise and Under-Voltage Protection
- Conservative 25 Amps at 8.5 Volts
- Conservative 5 Amps at ±16 Volts
- Conservative Component Rating for Reliabi ty

THE COMPONENTS

- Fully Socketed
- Gold Plated Bus Connectors
- Only "B" Series 68XX Components Used
- Only Top Grade Logic Circuits Used
- Industrial Grade Components Throughout

THE MEMORIES

DM-64

- Field Proven
- Proprietary Memory Control Logic
- Fully Transparent Refresh
- Tested at 2.5 MHz Operation

DM-512

- 512K Bytes on a Single S-64 Board
- 16 Bit Power and 8 Bit Compatibility
- Runs in Existing S-50 Systems where Physical Space Allows
- Full 24 Bit Addressing
- Fully Transparent Refresh

THE PRICES

Because of the variety of configurations possible, full pricing cannot be given. Representative prices are:

- **64K 6809 HELIX..... $2495**
- **256K 6809 HELIX..... $2895**
- **512K 6809 HELIX..... $3750**

HAZELWOOD COMPUTER SYSTEMS

907 E. Terra, O'Fallon, Missouri 63366 (314) 281-1055

Dealer and OEM Inquiries Invited. We support our Dealers.

Optima is a Trademark of Scientific-Atlanta